BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

10/815.731

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 13/00 — A1

(11) International Publication Number: WO 94/16387

(43) International Publication Date: 21 July 1994 (21.07.94)

(21) International Application Number: PCT/US94/00031

(22) International Filing Date: 3 January 1994 (03.01.94)

(30) Priority Data:
08/000,272 4 January 1993 (04.01.93) US

(71) Applicant: COACTIVE COMPUTING CORPORATION [US/US]; Suite 221, 1301 Shoreway Road, Belmont, CA 94002 (US).

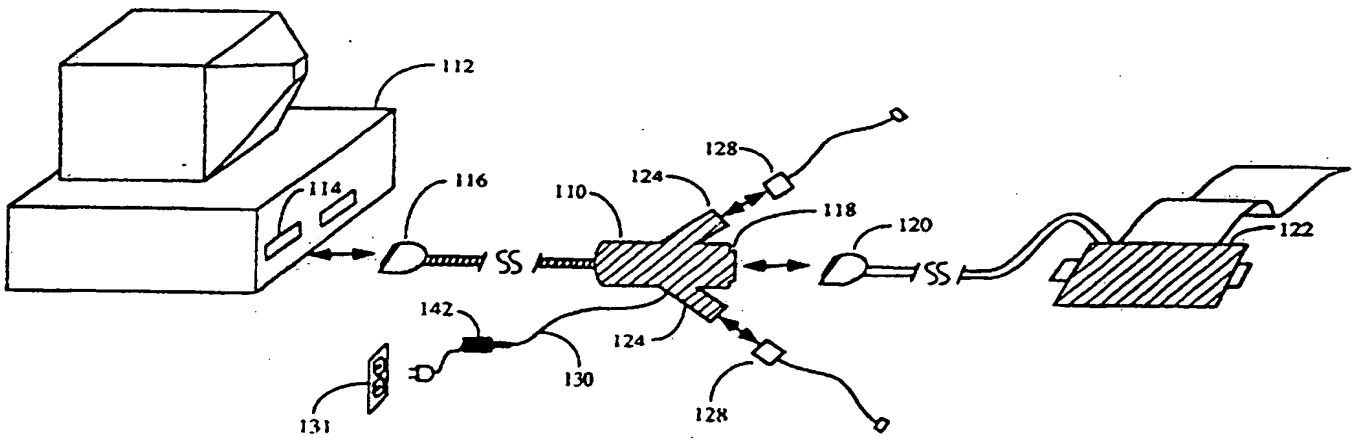
(72) Inventors: ROBINETT, Robert, S.; 485 Cotton Street, Menlo Park, CA 94025 (US). FOLEY, David, G.; 2671 Bryant Street, Palo Alto, CA 94306 (US). DYSON, Patrick; 1515 Alameda de las Pulgas, San Carlos, CA 94070 (US). WHEELER, Robert, B.; 24557 Pontiac Street, Hayward, CA 94544 (US). NEIMAN, Seth, D.; 1510 Los Altos Drive, Burlingame, CA 94010 (US). MASER, Scott, D.; 493 Fullerton Court, San Jose, CA 95111 (US). WOHNOUTKA, Robert, M.; 1533 Jasper Drive, Sunnyvale, CA 94087 (US).

(74) Agents: SALTER, James, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, 7th floor, Los Angeles, CA 90025 (US).

(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: COMPUTER INTERFACE APPARATUS FOR COMMUNICATING WITH A PERIPHERAL DEVICE AND NETWORK



(57) Abstract

A computer interface device (110) providing a networking and peripheral interface capability using an external port (114) of a host computer (112) is disclosed. The computer interface device (110) of the present invention provides an intelligent interface (110) between a standard external port (114) of a conventional computer (112) and both a peripheral device (122) and a computer network. The computer interface device (110) comprises a host interface (116) for connecting the computer interface device (110) to a host computer (112) via its external data interface. Control and data lines are received by the computer interface device (110) through the host interface (116). The control and data signals are passed to a control unit comprising switch logic for selectively switching the control and data lines received via the host interface (116) through to a peripheral device (122). The control logic may be used to alter the operation of the control signals and to communicate with a network via a pair of network ports (124) provided in the computer interface device (110).

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SI	Slovenia
SK	Slovakia
SN	Senegal
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

COMPUTER INTERFACE APPARATUS FOR COMMUNICATING WITH A PERIPHERAL DEVICE AND NETWORK

BACKGROUND OF THE INVENTION

1. Field Of The Invention

The present invention relates to the field of computer networks. Specifically, the present invention pertains to network interfaces for connecting computers and peripheral devices together thereby allowing communication between computers and peripheral devices.

2. Prior Art

Many different computer networks exist in the prior art. Some computer networks, such as Ethernet, require the use of a shielded cable for data transmissions. These cables are typically expensive and sometimes difficult to install. In addition, typical prior art networks often require specialized hardware installed within each computer system on the network. This network hardware is usually installed and configured by specially qualified network or computer system service personnel whose expertise is typically beyond the skill level of most computer system users. Thus, installation of these networks is typically an expensive operation. Often times, the installation of network hardware in a computer system leads to system failures that are difficult to trace and correct.

It is advantageous to use existing general purpose external input/output (I/O) hardware for the purpose of connecting a computer to a network. In this manner, a computer may be connected to a network without the use of specialized installation skills and personnel. One method is to use a standard parallel port on a computer for the purpose of interfacing with a network. The use of a parallel port is advantageous because most computer systems, particularly small personal computers, are manufactured with a parallel port as standard external interface equipment.

There are a number of problems with using parallel ports for a network interface. First, the parallel port may already be in use by a peripheral device such as a printer. If the parallel port is used as a network interface, the ability to communicate with a peripheral device originally connected to the parallel port cannot be lost in the process. Thus, a network interface device must be capable of communicating with both a network and a peripheral device coupled to a computer through a parallel interface.

A second problem in using a parallel port for network interface is the lack of standardization across the parallel interface. Depending upon the type and age of a particular computer system, the parallel port interface is implemented in slightly different ways. In some systems, the parallel port is a data output-only interface intended strictly for receive-only devices such as printers or plotters. Clearly, the use of a parallel port for network communications is only possible across a bi-directional interface. A computer system providing a data output-only parallel port, therefore, presents significant problems for a network interface device using the parallel port. Other computer systems may provide a bi-directional interface; however, these interfaces may not be implemented in a consistent and standardized manner. For example, some parallel interface implementations provide STROBE and ACKNOWLEDGE (ACK) control signals for strobing data through the parallel interface in either an input or an output direction. Other implementations may not provide a strobed interface protocol on the input of data to the computer system. Still other computer systems may not use the same control signals or may implement the interface control signals using a different and incompatible timing protocol. Thus, the many variations in the implementation of a parallel port interface makes it very difficult to use a parallel port for network communications.

A system for connecting a plurality of computers and computer peripherals into a network using two unshielded lines of the type included in a telephone cable is disclosed in U.S. Patent No. 5,003,579, by Reese M. Jones (hereinafter the Jones Patent). The

Jones Patent discloses a network connector providing simple, effective, and low-cost interconnection among similar or diverse microcomputers and peripherals and over long-running lengths of ordinary telephone cable. The Jones Patent, however, does not provide a means for interpreting and reconfiguring the interface control signals to/from a host computer. This manipulation of control signals is necessary in order to provide a network interface that is compatible with computer systems implementing an I/O interface in varying and typically incompatible ways. Further, the Jones Patent does not disclose a network interface device having a means for optionally switching information through to a peripheral device such as a printer. Such a capability is necessary to provide a network interface device that may be installed using an existing computer system interface that is already allocated for a different purpose.

Other low cost computer networks exist in the prior art such as the AppleTalk™ network developed by Apple Computer™, Inc. The AppleTalk network includes a network interface device that plugs into an external I/O interface of a host computer and transfers data across a shielded cable network. In a manner similar to the Jones Patent, the AppleTalk network device does not provide a means for manipulating interface control signals or a means for switching data through to a peripheral device.

Thus, a better computer interface device is needed.

SUMMARY OF THE INVENTION

The present invention is a computer interface device providing a networking and peripheral interface capability using an external port of a computer. The computer interface device of the present invention provides an intelligent interface between a standard external port of a conventional computer and both a peripheral device and a computer network. The computer interface device comprises a host interface for connecting the computer interface device to a host computer via its external data interface. In the preferred embodiment, the external data interface is a parallel port. Control and data lines are received by the computer interface device through the host interface. The control and data signals are passed to a control unit comprising switch logic for selectively switching the control and data lines received via the host interface through to a peripheral device. The control and data lines are also routed through the switch logic to control logic within the computer interface device. The control logic may be used to alter the operation of the control signals and to communicate with a network via a pair of network ports provided in the computer interface device. The control logic is used in cooperation with a processor within the computer interface device to control the operation of the switch logic and to control the state of the data and control signals passed across the host interface and network interfaces. The control logic includes an external port controller (EPC) circuit for controlling the signals received via the host interface and peripheral interface through the switch logic. The control logic also includes counters and timers, interrupt control logic, and memory control logic. The computer interface device also includes a read only memory (ROM) for storing non-volatile data for use during initial program loading of the computer interface device.

It is, therefore, an object of the present invention to provide a computer interface device having means for communicating with both a network and a peripheral device. It is a further object of the

present invention to provide a computer interface device wherein communication with a peripheral device may be selectively enabled or disabled. It is a further object of the present invention to provide a computer interface device wherein communication with a host computer is performed using a different set of control signals than used for communicating with a peripheral device. It is a further object of the present invention to provide a computer interface device wherein a host computer interface selectively provides a 2-bit, 4-bit, or 8-bit data interface. It is a further object of the present invention to provide a computer interface device wherein communication with a host computer is performed bi-directionally using the same handshake protocol. It is a further object of the present invention to provide a computer interface device wherein communication with a host computer is performed bi-directionally using a handshake protocol that is timing independent. It is a further object of the present invention to provide a computer interface device wherein communication with a host computer is performed bi-directionally using a polled or interrupt driven protocol. It is a further object of the present invention to provide a computer interface device wherein communication with a host computer employs a selection protocol whereby the erroneous transfer of data is eliminated. It is a further object of the present invention to provide a computer interface device wherein communication with a peripheral device may be controlled to allow the peripheral device to be selectively shared on a network or dedicated to local use by a single computer. It is a further object of the present invention to provide a computer interface device wherein communication between a host computer and a peripheral device can be performed in a pass-through mode whereby communication between the host computer and the peripheral device is unmodified and unimpeded even though the computer interface device may be unpowered, powered but uninitialized, or powered and initialized. It is a further object of the present invention to provide a computer interface device wherein a host computer must be able to communicate with a

peripheral device as if a computer interface device were not present. Furthermore, this must occur even when no power is supplied to the computer interface device. It is a further object of the present invention to provide a computer interface device wherein a host computer must be able to communicate with a computer interface device without interfering with or receiving interference from a peripheral device. It is a further object of the present invention to provide a computer interface device wherein a computer interface device must be able to communicate with a peripheral device without interference from a host computer. It is a further object of the present invention to provide a computer interface device wherein control logic within the computer interface device must be protected against electro-static discharge (ESD) and electrical surges or spikes which could be caused by connecting the computer interface device to a powered-up host computer or a powered-up peripheral device. It is a further object of the present invention to provide a computer interface device wherein switch logic can be configured such that signals can be driven from either side of the interface. It is a further object of the present invention to provide a computer interface device wherein the computer interface device interfaces with the host computer or peripheral device over a parallel, serial, or infrared data link.

The realization of these and other objects and advantages of the present invention will become apparent as presented and described in the following detailed description of the preferred embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates the computer interface device of the present invention as connected with a host computer and a peripheral device.

Figure 2 illustrates a network topology using the computer interface device of the present invention.

Figure 3 is a block diagram illustrating the internal hardware of the computer interface device of the present invention.

Figures 4A, 4B, 4C, and 5 illustrate the switching logic of the computer interface device of the present invention.

Figure 6 illustrates the network interface of the computer interface device.

Figures 7-18 illustrate the content of the programmable registers in the preferred embodiment.

Figures 19 and 20 illustrate the state of the Data Director multiplexers shown for each mode in which the Data Director operates.

Figures 21-23 illustrate the relationship between the registers and the host interface signals generated in each of the three Data Director modes of operation.

Figure 24 is a table illustrating the relationship between the operating modes of Data Director and the state of the control and data lines.

Figures 25 and 26 are timing diagrams illustrating the data transfer protocol of the computer interface device of the present invention.

Figure 27 is a timing diagram illustrating the selection sequence used in the present invention.

Figure 28 is a block diagram illustrating the software of the computer interface device of the present invention.

Figures 29-33 are flowcharts illustrating the processing logic of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is a computer interface device providing a networking and peripheral interface capability using an external port of a host computer. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be apparent to one of ordinary skill in the art that these specific details need not be used to practice the present invention. In other instances, well known structures, circuits, and interfaces have not been shown in detail in order not to unnecessarily obscure the present invention.

Referring to Figure 1, the present invention is shown in an environment in which the invention typically operates. The computer interface device of the present invention provides an intelligent interface between a standard external port of a conventional computer and both a peripheral device and a computer network. In the preferred embodiment, the external port is a parallel port; however, other alternative embodiments of the external port are described below. Referring to Figure 1, a computer 112 is shown with a standard external port interface 114. In a conventional configuration, a peripheral device (i.e. printer 122) is connected directly to computer 112 via cable connector 120 and external interface port 114. In this configuration, computer 112 is a standalone computing device with a dedicated peripheral resource coupled to it.

It is convenient to expand the conventional computing system environment to provide a means for connecting computer 112 with other computers in a network. It is also convenient to improve the conventional computer system configuration by allowing peripheral device 122 to be shared by other computers in a network.

The present invention is a network and peripheral device interface unit providing network access for both computer 112 and peripheral device 122. The computer interface device 110 comprises circuitry contained within a housing and powered by an AC/DC power converter 142. Conventional circuitry present

within the AC/DC power converter 142 converts 110 volt AC power 131 or any other standard power source to approximately a 9 volt DC source for driving the circuitry within computer interface device 110. Computer interface device 110 also comprises a host interface 116 which is implemented in the preferred embodiment using a standard DB-25 male parallel port connector for connecting computer interface device 110 with the parallel port 114 of computer 112. In an alternative embodiment, host interface 116 comprises a serial port connection for coupling the computer interface device 110 with host computer 112 across a serial interface. In yet another alternative embodiment, host interface 116 comprises an infrared port connection for coupling the computer interface device 110 with host computer 112 across a wireless infrared data link. Other conventional data communications methods may equivalently be used. In general, parallel, serial, and infrared data communications methods are well known to those of ordinary skill in the art.

As will be described more fully below, the interface signals provided by connector 116 and used by computer 112 and computer interface device 110, are provided in several configurations in the preferred embodiment. In one configuration, a standard IBM™ PC parallel port is supported. In this configuration, an 8 bit unidirectional data transfer interface is provided from computer 112 to computer interface device 110. In a second configuration, an 8 bit unidirectional parallel data transfer from computer 112 to computer interface device 110 is combined with an unstrobed read (i.e. a data transfer from computer interface device 110 to computer 112). In a third configuration, an IBM™ PS/2 type parallel port is supported. This parallel interface comprises an 8 bit bidirectional parallel data transfer interface with data strobed in both directions. Another parallel interface supported by the computer interface device 110 of the present invention is an 8 bit unidirectional parallel data transfer interface from computer 112 to computer interface device 110 wherein not

all status lines are supported. Such an interface is commonly used by devices compatible with Tandy™ Corporation brand computers.

Computer interface device 110 also includes a peripheral interface 118 for connecting peripheral device 122 with computer interface device 110 via connector 120. In the preferred embodiment, these peripheral devices comprise printers, plotters, and other devices typically operating mainly as data receivers. It will be apparent to those of ordinary skill in the art that peripheral devices other than printers may benefit from the operation of the present invention. In the preferred embodiment, connector 118 is a standard DB-25 female connector. The external port signals provided by peripheral interface 118 are the same as the signals provided by host interface 116. In an alternative embodiment, peripheral interface 118 comprises a serial port connection for coupling the computer interface device 110 with a peripheral device 122 across a serial interface. In yet another alternative embodiment, peripheral interface 118 comprises an infrared port connection for coupling the computer interface device 110 with peripheral device 122 across a wireless infrared data link. Other conventional data communications methods may equivalently be used. In general, parallel, serial, and infrared data communications methods are well known to those of ordinary skill in the art. As will be shown below, the manipulation and control of signals across host interface 116 and peripheral device 122 for the various peripheral port configurations and operating modes of computer 112 is a formidable problem.

In some situations, an external port of a first type (e.g. parallel) may be used to interface with a host computer at the same time an external interface of a second type (e.g. serial) is used to interface with a peripheral device. In these situations, signals of the first type of interface must be converted to signals compatible with the second type of interface and vice versa. This signal conversion is performed in the computer interface device 110 of the present invention using signal conversion techniques well known to those of ordinary skill in the art.

Computer interface connector 110 also comprises network interfaces 124. In the preferred embodiment, these interfaces are RJ-11 type interface connectors. These network interfaces are used to connect computer interface device 110 to a network via RJ-11 connector 128 and a standard telephone wire. In this manner, computer 112 is connected through computer interface device 110 to a computer network through interfaces 124 and to a peripheral device 122 through interface 118.

Referring now to Figure 2, a plurality of computer interface devices are shown in a typical computer network configuration. Computer interface device 210 is shown coupled to computer 212 via the host interface 213 of computer interface device 210. Peripheral device 214 (i.e. a printer device) is shown coupled to the computer interface device 210 via the peripheral interface and connector 215. In a similar manner, network interface device 216 is shown coupled to computer 218, via the host interface, and to peripheral device 220 via the peripheral interface. Using a standard modular wire 217, computer interface device 210 is coupled to computer interface device 216. In this configuration, computer 212 may communicate with computer 218 via computer interface device 210, network cable 217, and computer interface device 216. The present invention provides a means for automatically terminating network interfaces such as network interface 233 that remain unconnected.

The computer interface device of the present invention also provides a means for configuring the communications between a host computer and a peripheral device in either a local or shared mode. In a local mode, a computer interface device only allows a connected peripheral device to communicate with the host computer connected to the computer interface device. In this mode, a peripheral device may be dedicated for use by one computer. Printer 214 illustrated in Figure 2 represents an example of a peripheral device operating in a local mode.

Using the computer interface device of the present invention, peripheral devices may also be selectively operated in a shared

mode. In a shared mode, a peripheral device may be accessed and used by any of the computers on a computer network, such as the network illustrated in Figure 2. Printer 220 represents an example of a peripheral device operating in a shared mode. For example, computer 212 may access shared peripheral device 220 by transferring command information and data through computer interface device 210, network cable 217, computer interface device 216, and finally to peripheral device 220. As will be shown below, the computer interface device 216 of the present invention controls the operation of a local or shared mode at the request of a host computer to which it is connected. The computer interface device 216 of the present invention may operate in a shared mode even when the host computer 218 is not powered.

The computer interface device 222 illustrated in Figure 2 represents an example of the present invention in operation without a peripheral device coupled to it. In this configuration, the computer interface device 222 is used purely for coupling a host computer to a computer network.

Computer interface device 226 represents an example of the present invention in use with a non-printer peripheral device 230. In this example, the present invention provides a means for interfacing with a large range of peripheral devices including various types of printers, plotters, special purpose equipment, or other devices which are compatible with a particular data communication interface.

In another example of the operation of the present invention, computer interface device 226 is shown coupled to a standard AppleTalk compatible connector 232. Such a connector 232 is well known and commonly available in the prior art. Using the AppleTalk compatible connector 232, a host computer (such as a Macintosh™ computer manufactured by Apple Computer™, Inc.) may be connected to a computer network via connector 236 and AppleTalk compatible connector 232. In this manner, various types of host computers (such as IBM™ compatible computers, Apple™ compatible computers, or other computers compatible

with a common network protocol) may communicate and share information using the computer interface device of the present invention. Similarly, a printer 244 may be coupled to the network using a standard AppleTalk compatible connector 242 as shown in Figure 2. The Apple™ compatible computer 234 or any of the IBM™ compatible computers may thereby access printer 244. The Apple™ compatible computer 234 may also access shared printer 220 and other computers (i.e. 212, 218, 224, and 228) on the network. In a similar manner, devices other than printers may also be coupled to the network. These other devices include, for example, a network modem or network facsimile machine. It will be apparent to one of ordinary skill in the art that other network compatible devices may also be coupled to the network and shared by network computers.

Referring now to Figure 3, a block diagram of the computer interface device 520 of the present invention is illustrated. The computer interface device 520 comprises host interface 522 for connecting the computer interface device 520 to a host computer 510 via its external data interface 512. The signals traveling between host computer 510 and computer interface 520 across the external interface 512 comprise nine control/status lines and eight bits of data in the preferred embodiment. The nine control lines include a STROBE signal, an INITIALIZATION (INIT) signal, an ERROR signal, a SELECT_IN (SLCT_IN) signal, an EMPTY signal, a BUSY signal, a SELECT signal, an ACKNOWLEDGE (ACK) signal, and an AUTOFEED (AF) signal. The STROBE signal is used to latch data into a peripheral device. The INIT signal or initialization strobe is used to reset the peripheral device. The ERROR signal is a flag that indicates that the peripheral device is not functioning properly. The SELECT_IN signal is used to set a peripheral device in an on-line or off-line mode. Transmission of data to a peripheral device is possible only when this line is active. The EMPTY signal is used with printing devices to indicate a paper empty or paper out condition. The BUSY signal is a flow control line that indicates that the peripheral device is unable to handle

additional input. The SELECT signal is a flag indicating whether the peripheral device is on-line or off-line. The ACKNOWLEDGE or ACK signal is used by a peripheral device to acknowledge the receipt of data. The AUTOFEED signal, in its conventional use, indicates that the peripheral device (printer) should automatically line feed at the end of every line on receipt of a carriage return (CR) character.

These control and data lines are received by computer interface device 520 through host interface 522. The control and data signals received by host interface 522 are passed to control unit 523 via interface 526. Control unit 523 comprises switch logic 524 for selectively switching the control and data lines received by host interface 522 through to peripheral device 571 via lines 530 and peripheral interface 528. The control and data lines are also routed through switch logic 524 to control logic 532 via interface 536. Control logic 532 is used in cooperation with processor (CPU) 538 to control the operation of switch logic 524 and to control the state of the data and control signals passed across host interface 522 and peripheral interface 528. Control logic 532 comprises external port controller (EPC) circuit 533 for controlling the signals received via host interface 522, peripheral interface 528, and switch logic 524. Control logic 532 also includes a read only memory (ROM) 542 for storing non-volatile data for use during initial program loading of computer interface 520. The design and operation of EPC 533 and the initial program loading of the computer interface device 520 is described in detail below.

Control logic 532 also includes counters and timers 581, interrupt control logic 582, and memory control logic 583. Counters and timers logic 581 includes a net idle timer. The net idle timer reports the number of microseconds that the AppleTalk telephone wire network has been idle. Because the AppleTalk protocols employed on the network require that the network be idle for at least 400 usecs. before a packet is transmitted, net idle timer eliminates this 400 usec. wait under the typical condition of an idle network. The net idle timer starts counting as soon as the network

becomes idle and continues counting as long as the network stays idle. When activity is detected on the network, the net idle timer is reset. Instead of having to wait a full 400 usecs. for each transmission over the network, the CPU 538 can access the net idle timer to determine how much time (if any) the CPU 538 must wait to satisfy the 400 usec. delay time. Typically, the CPU 538 waits, if at all, for a significantly shorter time than the full 400 usecs. This permits higher utilization of the network and CPU bandwidth under most normal usage conditions. Memory control logic 583 is used to control access to RAM 544.

Control logic 532 also includes a CPU interface 540 through which address, data, and control lines are routed between control logic 532 and CPU 538. In the preferred embodiment, CPU 538 is a 68EC000 embedded controller with a 16 bit data bus and a 24 bit address bus. Such an embedded controller is available from Motorola™ Corporation. Control logic 532 also includes dynamic memory interface circuitry and a Data Director circuit 545 with a two channel DMA (direct memory access) controller with which control logic 532 may access random access memory (RAM) 544 for the storage and retrieval of data to be transferred across host interface 522 on a DMA channel. The Data Director and DMA controller 545 is described in more detail below.

In the preferred embodiment, computer interface 520 also includes an interface 562 for communicating with a read only memory (ROM) 558 coupled externally to the control unit 523. In this manner, additional non-volatile memory may be provided for computer interface device 520. Control logic 532, in the preferred embodiment, also includes an interface 560 to any conventional eight bit peripheral device. Control logic 532 also includes a network interface 593 which is coupled through high level data link control logic (HDLC) 580 to a serial network driver 550 via line 554. HDLC 580 frames data packets and formats signals that are transmitted on the computer network. HDLC 580 logic is well known to those of ordinary skill in the art. Serial network driver 550 is coupled to two network ports/interfaces 4314 and 4316 as

shown in Figures 3 and 6. Network ports 4314 and 4316 are used to connect computer interface device 520 to a computer network as shown in Figure 2. In the preferred embodiment, the computer network uses standard telephone wire network interconnections with an AppleTalk network protocol. An AppleTalk network protocol itself is well-known to those of ordinary skill in the art. In an alternative embodiment, the computer interface device of the present invention may also be used with an Ethernet network or other conventional network protocols. Network ports 4314 and 4316 correspond to interfaces 124 illustrated in Figure 1. Peripheral interface 528 corresponds to interface 118 shown in Figure 1. Host interface 522 corresponds to host interface 116 illustrated in Figure 1. The routing and control of the control and data signals received through host interface 522 will now be described in reference to Figures 4A, 4B, 4C and 5.

Referring again to Figure 3, switch logic 524 is designed to provide at least the following capabilities that are necessary for the successful use of any computer interface device 520 attached to an external port of a host computer 510:

1. Host computer 510 must be able to communicate with peripheral device 571 as if computer interface device 520 were not present. Furthermore, this must occur even when no external power is supplied to computer interface device 520.
2. Host computer 510 must be able to communicate with computer interface device 520 without interfering with or receiving interference from peripheral device 571.
3. Computer interface device 520 must be able to communicate with peripheral device 571 without interference from computer 510.
4. Control logic 532 must be protected against electro-static discharge (ESD) and electrical surges or spikes which could be caused by connecting computer interface device 520 to a powered-up host computer 510, or a powered-up peripheral device 571.
5. Because some widely-used peripheral devices 571 drive the signal lines 530 in non-standard directions, switch logic 524 must

be implemented in such a way that signals can be driven from either side.

Referring now to Figures 4A, 4B, and 4C, a switching circuit of switch logic 524 is illustrated. This switching circuit is present on each of the signal lines of lines 526, 536, and 530. These signal lines connect switch logic 524 with host interface 522, control logic 532, and peripheral interface 528, respectively. In the preferred embodiment, there are 17 control and data signal lines coupled to each of three ports of switch logic 524 using this switching circuit. In the preferred embodiment, these control and data lines include AUTOFEED, ACK, SELECT, BUSY, EMPTY, SELECT_IN, ERROR, INIT, STROBE, and eight data bits lines. Figure 4A shows this switching circuit that is used for each of the signal lines other than the AUTOFEED line. Figure 4B shows the switching circuit used for the AUTOFEED line including the addition of a single diode for accommodating printers that force the AUTOFEED line into an asserted or deasserted state using a toggle switch or the like. Figure 4C shows the switching circuit with communication enabled between host computer 410 and peripheral interface 414. The switching circuit of switch logic 524 provides all five of the capabilities listed above.

Referring to Figure 4A, the switching circuit of switch logic 524 comprises a switch 422 (e.g. one section of a single pole switch SPST which is commonly available from Toshiba, Inc.as a switch designated 4066) that in its normally closed state carries signals in both directions between host computer 410 and peripheral interface 414 via lines 416 and 418. Another switch 424 (e.g. one section of an SPDT switch commonly available from Toshiba, Inc. as a switch designated 4053) allows control logic 412 to connect to either host computer 410 via lines 420 and 416 or peripheral interface 414 via lines 420, 428, and 418. Another section 425 of the SPDT switch allows control logic 412 to be isolated from host computer 410 and peripheral interface 414 when control logic 412 is unpowered. The PERIPHERAL DISCONNECT signal on control line 426 controls switch 422. The COMPUTER INTERFACE DIRECTION signal

on control line 430 controls switch 424. The COMPUTER INTERFACE ENABLE signal on control line 432 controls switch 425.

Referring to Figure 4B, the switching circuit used for the AUTOFEED line comprises the same circuit as that described in Figure 4A, except a diode 440 is added on line 418 between switch 422 and peripheral interface 414. The addition of a single diode 440 accommodates printers that force the AUTOFEED line into an asserted or deasserted state using a toggle switch or the like.

Referring to Figure 4C, the switching circuit described in Figure 4A is shown, except that the switches 422 and 425 are closed. This is accomplished using control lines 426 and 432. With these switches closed, communication is enabled between host computer 410, control logic 412, and peripheral interface 414.

Each switch of switch logic 524 comprises an analog CMOS (complimentary metal oxide semiconductor) switch used for selectively controlling a single control or data line received from host interface 522. An analog switch of this type is well known and commonly available from Toshiba, Inc as referenced above.

As shown in Figures 4A-4C, each analog switch has a signal line (e.g. 416, 418, and 420) input and a control line (e.g. 426, 430, and 432) input. Each control line may be used to selectively activate the switch to close or open the switch. In Figure 4A, switches 422 and 425 are shown in the open position as achieved if control signals 426 and 432 are activated. If control signals 426 and 432 are deactivated, switches 422 and 425 are closed to complete a circuit on signal lines 418 and 420. In this manner, signal transmission across the signal lines may be enabled or disabled using the control lines.

Referring again to Figure 3, computer interface device 520 includes low power control logic 566. In order to provide the first capability listed above (i.e. communication with an unpowered computer interface device 520), a self-powering capability is provided by low power control logic 566. Low power control logic 566 is coupled to host interface 522 on line 568. Alternatively, low

power control logic 566 can be coupled to peripheral interface 528 on a second set of signal lines (not shown). Low power control logic 566 uses electric current taken from the host interface 522 signal lines STROBE and INIT from host computer 510 and/or the ACK, BUSY, and SELECT lines from peripheral device 571 to provide a +7 volt power supply voltage to the CMOS switches of switch logic 524. This electric current ensures that switch logic 524 will pass signals between host interface 522, control logic 532, and peripheral interface 528 without any loss of signal voltage level.

In order for data transfers between host computer 510 and peripheral device 571 to begin, host computer 510 must perform a selection sequence with computer interface device 520. This sequence is designed to minimize the possibility that computer interface 520 would incorrectly disconnect a peripheral device 571 from computer 510 during the course of normal data transfer between them. Any errors detected by computer interface device 520 during selection causes it to return switch logic 524 to its default state. A timing diagram of the selection sequence is illustrated in Figure 27. A detailed description of the selection sequence is presented below in connection with Figure 27.

Referring to Figure 5, the interconnection of signal lines used by the host interface data transfer protocol of the present invention is illustrated. As shown, signal lines 4140 connect host interface 522 to control logic 532 through switches 4180 within switch logic 524. These same signal lines are also connected to peripheral device 528 through switches 4180, which are described above in connection with Figures 4A-4C.

A data transfer from host computer 510 to control logic 532 is initiated by host computer 510. Host computer 510 outputs eight bits of data on data lines 0:7 of signal group 4100 and asserts the AUTOFEED signal on a line of signal group 4120. The control logic 532 then reads the data on data lines 0:7 in signal group 4140 and acknowledges its receipt by asserting the ACK signal of signal group 4120.

Similarly, a transfer from control logic 532 to host computer 510 is initiated by host computer 510. Host computer 510 asserts the AUTOFEED signal on a line of signal group 4120. The control logic 532 then outputs the data on the status lines and acknowledges that the output data is valid by asserting the ACK signal of signal group 4120. In a two bit mode, the data is output on the BUSY and PAPER ERROR lines of signal group 4140. In a four bit mode, the data is output on the BUSY, PAPER_ERROR, INIT, and SELECT_IN lines of signal group 4140. When connected to a host interface 522 that can disable its outputs on data lines 0:7, control logic 532 can then drive all eight bits of those control lines in an eight bit mode. It will be apparent to those of ordinary skill in the art that data may be sent to the host computer 510 in any configuration of lines provided between control logic 432 and host computer 510.

In the preferred embodiment, the AUTOFEED signal is used by a master device to initiate a data transfer to a slave device. In establishing communication between the host computer 510 and the computer interface device 520, it is necessary to disconnect the AUTOFEED signal line from peripheral device 571 before all the other signals in signal group 4200 are disconnected and communication with the computer interface device 520 is established. For this purpose, a separate AUTOFEED DISABLE control line in signal group 4160 is provided. Furthermore, in order to ensure that a powered-off printer connected to peripheral interface 528 does not short out the AUTOFEED line in signal group 4200 and therefore make it impossible for the host computer 510 to control the AUTOFEED signal line in signal group 4120, a diode 440 is provided in series on the AUTOFEED line as shown in Figure 4B. When a printer is connected as the peripheral device 571, it must never be allowed to see the change in AUTOFEED or it may insert unwanted blank lines on the pages it is printing.

The present invention provides logic in control logic 532 for performing data transfers between host computer 510 and computer interface device 520 in a polled mode. The external port

controller (EPC) 533 in control logic 532 is used for this purpose. When EPC 533 operates in polled mode (the default after power-up), the CPU 538 individually sets up each of three groups of signals used in transfers with host interface 522 or peripheral interface 528. Each of these signal groups can be separately set up to be inputs or outputs as required by the direction of transfer, width of the data path, and other capabilities of the computer interface device 520. The signals grouped in this manner include: Data Bits [0:7], Control Signals (STROBE, INIT, SELECT_IN, and AUTOFEED), and Status Signals (ACK, BUSY, SELECT, ERROR, PAPER_EMPTY) Separate registers in EPC 533 provide access for the CPU 538 to read these signals, or write them when a signal group's output is enabled. Registers within EPC 533 are described below.

In addition to a polled data transfer mode, control logic 532 provides logic for direct memory access (DMA) data transfers. There are two modes for EPC 533 DMA operation. In DMA Master Mode, EPC 533 initiates each transfer to a slave device by outputting data on Data Bits [0:7] of signal group 4140 and asserting the AUTOFEED signal. The slave device signals that it has received the data by asserting the ACK signal, at which point the Master device releases the AUTOFEED signal to indicate the end of the byte transfer.

In a DMA slave mode, the slave device obtains one byte of data from RAM 544 and performs one, two, or four handshakes that are required to transfer eight bits of data to or from the master device. Each handshake is initiated by the master device asserting AUTOFEED, after which the slave device then posts the data on the output lines and asserts the ACK signal. The master device then deasserts AUTOFEED to signal that it has accepted the data, and the slave device then deasserts ACK to signal that it is ready for the next transfer cycle. When set up in a two bit or a four bit mode, the EPC 533 selects the correct bits to output on the status lines during each of the transfer cycles. In a two bit mode, there are four transfer cycles for each byte of RAM 544 data. In a four bit

mode, there are two transfer cycles for each byte of RAM 544 data. In an eight bit mode, there is one transfer cycle and the data is output on data lines [0:7]. In an eight bit mode, the master device must be capable of disabling its data bit outputs so that it can read the slave device data when this mode is used.

In order to minimize interference between computer interface device 520 and other devices attached to the external port signal lines, software running on host computer 510 must execute a selection sequence with computer interface device 520 prior to any data transfer in either direction. The selection sequence is described below in connection with Figure 27. When the selection sequence is successfully completed, computer interface device 520 sets switch logic 524 so that all signal lines from host computer 510 are connected to control logic 532 and all signal lines on peripheral interface 528 are disconnected. During this Selected state, data transfers between the control logic 532 and host computer 510 can be performed by CPU 538 in a Polled Mode. Alternatively, data transfers between the control logic 532 and host computer 510 can be performed under the control of the DMA Controller and Data Director 545 in a DMA Slave Mode. Upon detection of a data transfer error, or in response to a command from host computer 510, control logic 532 can return to an Idle state, where all host interface 522 signal lines are connected to peripheral interface 528, and all of the signal lines coupled to control logic 532 are inputs until another selection sequence is detected.

Computer interface device 520 can also communicate with another computer interface device which is attached to peripheral interface 528. This communication between two computer interface devices is accomplished in Polled Mode using CPU 538 or in DMA Master Mode using the Data Director and DMA controller 545.

When commanded by host computer 510 over host interface 522 or another computer interface device 520 over serial network driver 550, computer interface device 520 can also transfer character data to a printer which is connected to peripheral interface 528. To perform this operation, switch logic 524 is set so

that host interface 522 is disconnected from peripheral interface 528 and control logic 532 is connected to peripheral interface 528. This transfer can only be performed using Polled Mode in the preferred embodiment. To avoid having the CPU 538 poll the BUSY signal line from the printer, that line is also carried directly to control logic 532, where it can be enabled to generate an interrupt of CPU 538 when BUSY is low.

The EPC 533 includes a capability of accumulating a checksum of the data being transferred across the host interface 522. With the assistance of the CPU 538, this checksum provides a level of integrity for the communication link.

The EPC 533 further includes an interrupt generation capability through which it can request attention from the CPU 538. Interrupt sources include the STROBE, AUTOFEED, INIT, ACKNOWLEDGE and BUSY interface signals as well as a DMA terminal count flag. Interrupts are controlled by interrupt control logic 582 within control logic 532.

The EPC 533 further includes logic for manipulating switching control signals 4160 illustrated in Figure 5. These control signals 4160 are used to selectively couple/decouple signals between host interface 522, peripheral interface 528, and control logic 532 as specified by a set of registers within EPC 533. These registers are described in the following sections.

The External Port Controller (EPC) 533 contains eleven software accessible and programmable registers in the preferred embodiment. These registers, described below, provide a means by which software executing in CPU 538 can control the operational modes of the EPC 533, as well as data and status ports for communication across the host interface 522.

Referring to Figures 7-18, the eleven registers in the EPC 533 of the preferred embodiment are illustrated. Figure 7 illustrates the Acknowledge Control/Status Register. The Acknowledge signal is a handshake line that occurs in response to an AUTOFEED or STROBE initiated transfer. A STROBE initiated transfer is a transfer from the host computer 510 through the peripheral

interface 528 to peripheral device 571 connected thereto. An AUTOFEED initiated transfer is a transfer between control logic 523 and host interface 522.

When computer interface device 520 is in master mode, it will initiate a data transfer by presenting data on the eight bit data bus (i.e. the Data lines 0:7 of signal set 4140) and strobing the data with either the STROBE line, when a peripheral device is receiving the data transfer, or the AUTOFEED line when a network device is driving the data transfer to another computer interface device. Computer interface device 520 expects to see a target device (i.e. a device with whom a data transfer is conducted) recognize the data transfer by asserting the ACKNOWLEDGE signal line. In master mode, the ACKNOWLEDGE signal is an input to the computer interface device 520. In slave mode, computer interface device 520 will respond to STROBE or AUTOFEED initiated data transfers by asserting its ACKNOWLEDGE line. In this mode, the ACKNOWLEDGE signal is an output. Note that in order to drive the ACKNOWLEDGE signal, the ACK DISABLE signal of signal set 4160 must be activated. The ACKNOWLEDGE signal defaults to an input at power up .

Figure 8 illustrates the Autofeed Control/Status Register. Depending on the mode of the EPC 533 (Master or Slave mode), the AUTOFEED signal is either a data transfer strobe or a status line. When a peripheral device is the destination of a data transfer, the computer interface device 520 will output this bit (AUTOFEED) in the conventional manner. In a conventional use, the AUTOFEED signal indicates the printer should automatically linefeed whenever a carriage return character is received. This signal line may be pulled high or low by some printers. A diode 430 is inserted in the AUTOFEED line as shown in Figure 4A to prevent the AUTOFEED line from being forced high or low by these printers.

With the EPC 533 in master mode, the AUTOFEED line will be output through peripheral interface device 528 as a data transfer strobe. The target device responds with ACKNOWLEDGE. When

in slave mode, the computer interface device 520 will respond to AUTOFEED with its ACKNOWLEDGE signal; in slave mode, the AUTOFEED line is an input. This signal defaults to an input. To output on this line, the software executing in CPU 538 must set the Strobe Output Enable bit (i.e. bit 0 = 1 of the Switching Output Control Register).

Figure 9 illustrates the Switching Control Register. Activating bit 0 of this register (Computer interface device 520 Direction) causes Switch logic 524 to connect the control logic 532 to the Host interface 522. Deactivating bit 0 will connect the control logic 532 to the peripheral interface 528 and disconnect it from the host interface 522 (master mode).

A low value stored in the Acknowledge Disconnect #1 bit position of the Switching Control Register isolates the computer interface device 520 ACKNOWLEDGE line from the peripheral device's ACKNOWLEDGE line through the switch logic 524. It will also change the mode of an internal bidirectional buffer to an output mode. This will allow the computer interface device 520 to uniquely drive the ACKNOWLEDGE signal line. The power up default is high, thereby allowing the peripheral device's ACKNOWLEDGE signal to control the ACKNOWLEDGE signal back to host 510. The Acknowledge Disconnect #1 bit controls the state of the control logic 532 ACK DISABLE signal of signal set 4160.

A low value stored in the Host Interface/Peripheral Interface Disconnect bit position #2 causes the switch logic 524 to disconnect the peripheral interface 528 from the host interface 522. A high value will connect the host interface 522 to the peripheral interface 528. The power up default is high. This signal controls the control logic 532 BYPASS signal of signal group 4160 illustrated in Figure 5.

A low value stored in the Acknowledge Disconnect #2 in bit position #3 isolates the computer interface device 520 ACKNOWLEDGE line from the peripheral device 571 ACKNOWLEDGE line through the switch logic 524. Unlike

Acknowledge Disconnect #1 described above, the Acknowledge Disconnect #2 signal does not control the ACKNOWLEDGE bidirectional buffer. The power up default is high, allowing the peripheral device 571 ACKNOWLEDGE signal to control. This signal also controls the control logic 532 signal ACK DISABLE of signal set 4160.

A low value stored in the Autofeed Disconnect bit position #7 causes the switch logic 524 to isolate the computer interface device 520 AUTOFEED line from the peripheral interface 528 AUTOFEED line. This will allow the computer interface device 520 to uniquely drive the AUTOFEED line. The power up default is high, allowing the host computer 510 AUTOFEED signal to control. The Autofeed Disconnect signal alone controls the control logic 532 AUTOFEED DISABLE signal of signal set 4160.

Figure 10 illustrates the External Port Data Register. When DMA data transfers are disabled (External Port Mode Register bit 2 = 0), the External Port Data Register acts as a data port between the CPU 538 and the host interface 522. If DMA is enabled, all subsequent transfers between the host computer and the computer interface device 520 will be handled by the Data Director and DMA Controller 545 until a terminal count is reached or the DMA is terminated by the CPU 538. During this period, this register is unavailable and is directional (i.e. either receiving or transmitting data). In DMA Mode, all data transfer handshaking between the computer interface device 520 and the Host Computer is handled by an internal state machine (not shown) of Data Director 545. When not in DMA mode, to enable the output drivers, the Data Output Enable bit must be set (Switching Output Control Register bit 2 = 1).

Figure 11 illustrates the External Port Status Register. In DMA SLAVE mode, when the computer interface device 520 is responding to a Host Computer request, the Data Director 545 will control the status lines to communicate with the Host Computer in a manner corresponding to whether a 2-bit, 4-bit, or 8-bit data width is configured.

The BUSY signal is normally used as a flow control line (input to host computer) that indicates that the peripheral device (e.g. printer) is unable to handle additional input. For a data transfer to the host computer, the BUSY signal line will contain the least significant data bit. In a four-bit data mode, this signal will contain data bit #4 on the first data transfer and bit #0 on the second data transfer. In two-bit data mode, BUSY will contain bits #6, 4, 2 and 0 in the data transfer sequence.

The Paper Empty (PE) signal is normally used to indicate a Paper Empty condition in the peripheral device (e.g. printer). When used for data transfer to the host computer, PE contains the next to least significant bit. In four bit mode, this signal contains bit #5 on the first transfer and bit #1 on the second transfer. In two bit mode, PE contains the most significant bits #7, 5, 3 and 1 in the transfer sequence.

The Select (SLCT) signal is normally used as a flag that indicates that the peripheral device (e.g. printer) is on-line. In four bit mode, this signal contains bit #6 on the first data transfer and bit #2 on the second data transfer. In two bit mode, this bit is not used.

The ERROR signal is normally used as a flag that indicates that the peripheral device is not functioning properly. In four bit mode, this signal contains bit #7 on the first data transfer and bit #3 on the second data transfer. In two bit mode, this bit is not used.

Figure 12 illustrates the Peripheral Device Control (Strobes) Register. When in master mode, this register is used to drive the state of the three remaining strobe signals on the external interface (i.e. INITIALIZATION, SELECT_IN, and STROBE). In master mode, this register is readable and writable. When in slave mode, this register provides a view of the state of these lines on the interface, and as such is read only.

The INITIALIZATION (INIT) signal is used to reset a peripheral device (e.g. printer). The SELECT_IN (SLCT IN) signal is used to signal a peripheral device (e.g. printer) to accept

input data. Data transfer to the peripheral device (e.g. printer) is possible only when this line is low. Some peripheral devices, however, override this signal, such that it is not possible to use it to take a peripheral device off-line. The STROBE signal is used to latch data into peripheral device.

In slave mode, the power up default, the Peripheral Device Control (Strobes) Register is an input such that the CPU 538 may view the signals in this register in a read-only mode. To drive these signals, the Output Enable of Strobes bit (Switching Output Control Register bit 0 = 1) must be asserted.

Figure 13 illustrates the Switching Output Control Register. This register is used to enable or disable the driving of signals by the computer interface 520. A high value stored in the Output Enable of STROBES bit allows the computer interface device 520 to drive the STROBE signal lines: INIT, SLCT_IN, STROBE and AUTOFEED. A low value disables the output drivers. The power up default is low. Note that the AUTOFEED DISCONNECT bit of the Switching Control Register only controls the switch selection, but does not control the output. This bit does not cause the switching controls logic to isolate the AUTOFEED signal.

For the Output Enable of STATS bit, a high value stored in this bit allows the computer interface device 520 to drive the interface status signal lines: BUSY, PE, SLCT, ERROR and ACKNOWLEDGE. A low will disable the output drivers. The power up default is low. Note that ACKNOWLEDGE is controlled via the ACKNOWLEDGE DISCONNECT bits of the Switching Control Register or the Output Enable of STATS bit of this register. The ACKNOWLEDGE DISCONNECT#1 signal makes ACKNOWLEDGE an output while also isolating it through the switch logic 524 via the ACK DISABLE signal of signal set 4160. ACKNOWLEDGE DISCONNECT #2 controls the ACK DISABLE signal of signal set 4160 only, but does not enable the output buffer. This bit will enable ACKNOWLEDGE as an output along with the remaining status lines.

For the Output Enable of Data bit, a high value stored in this bit allows the computer interface device 520 to drive the interface data lines. The power up default is low.

For the Computer Interface Device 520 Connect bit, a high on this bit enables the computer interface device 520 to connect to the external interface selected by the computer interface device 520 Direction line described above. The power on default is low, thereby disconnecting the computer interface device 520 from both of the external interfaces.

Figure 14 illustrates the External Port Mode Register. The External Port Mode Register (EPMR) allows software executing in the CPU 538 to control the various enhanced modes of the external port controller 533. When DMA is enabled, the Transmit Mode select bit field allows the selection of three distinct Data Director 545 transmit modes. Figure 15 illustrates the transmit modes provided by the Data Director 545 and programmable in the External Port Mode Register.

Three DMA data output (transmit) modes are supported by the External Port Mode Register. The following transmit modes are provided in the preferred embodiment: 2-bit mode, 4-bit mode, and 8-bit mode. In 2-bit mode, automatic transfers of data from computer interface device 520 to the host computer 510 are performed on the PE and BUSY peripheral device status lines. This is the power up default mode. In 4-Bit Mode, automatic transfers of data from computer interface device 520 to the host computer 510 are performed on the BUSY, SLCT, PE and ERROR peripheral device status lines. In the 8-Bit Mode, automatic transfers of data from computer interface device 520 to the host computer 510 are performed on the host interface 522 bidirectional data lines (Data 0:7).

The DMA Enable bit allows the invocation of the DMA transfer function. Once a data transfer session has been established by CPU 538, the Data Director and DMA Controller 545 handle the remaining block transfers adhering to the AUTOFEED/ACKNOWLEDGE protocol without CPU 538

intervention. The Data Director logic 545 eliminates the need for CPU 538 intervention in the transfer between the host computer and computer interface device 520. The Data Director 545 performs all bit shuffling and handshaking necessary to transfer the data to/from the host computer. The power up default condition is disabled (0).

When DMA data transfer is enabled (see above), a low value stored in the DMA Direction Select bit position indicates that data is being read from host interface 522. A high value indicates that data is to be written to the host interface 522. The power up default is low to facilitate initial program load.

During DMA transfer mode, the Master/Slave Mode Select bit determines whether the computer interface device 520 is a master or slave on the host interface 522. When the Master/Slave Mode Select bit is set to a high value, the computer interface device 520 will assume the role of Bus Master, thereby driving the data and strobe signal lines while monitoring the status signal lines. When the Master/Slave Mode Select bit is set to a low value (the power on reset default), the computer interface device 520 is in slave mode which is defined as receiving the data and strobe signals and driving the status lines. When DMA transfer mode is not enabled, the computer interface device 520 is said to be in polled mode. In this mode, the Master/Slave Mode Select bit is meaningless.

The External Port Controller (EPC) Master Reset Request bit will reset all internal external port state machines and set the EPC 533 to its power up defaults when this bit is set to a value of one. This bit automatically clears itself. After reset, the EPC 533 will be in a power on state and must be completely re-initialized.

Figure 16 illustrates the mode definitions for the External Port Interrupt Enable Register. A high value set on a particular bit of the interrupt enable register allows the particular signal to generate a CPU 538 level three interrupt. A low on any particular bit will prevent that interrupt from occurring. A high to low transition on the STROBE, INIT, AUTOFEED, ACKNOWLEDGE, BUSY or DMA Terminal Count lines will generate an interrupt request.

Figure 17 illustrates the External Port Interrupt Status Register. Regardless of the state of the External Port Interrupt Enable Register (described above), the latched status of each interrupt source will be available for CPU 538 inspection using the External Port Interrupt Status Register.

Figure 18 illustrates the External Port Checksum Register. During DMA Transfer mode, the checksum logic will accumulate the longitudinal checksum of the data being transferred. Prior to the DMA transfer operation, the CPU 538 is expected to clear the checksum register by writing to it. The value written is irrelevant. The DMA logic 545 then performs a bitwise "exclusive OR" operation on every successive transfer datum with the previously accumulated result. At the end of DMA transmission, the register will contain the packet "checksum" for CPU 538 post processing. The checksum is not automatically transferred. It is expected to be transferred by the CPU 538 during the session closing sequence which occurs in polled mode.

Referring now to Figures 19 and 20, the sequencing of data used by Data Director 545 is illustrated. Data Director 545 uses a cascaded series of multiplexers to provide a selected data width for data transfers to the host interface 522. Figure 19 illustrates the state of the Data Director 545 multiplexers shown for each mode in which the Data Director 545 operates. A value of '1' in Figure 19 indicates that the specified multiplexer shown in Figure 20 will select the upper wire, while a value of '0' indicates the selection of the lower wire.

As shown in Figure 19, the data is transmitted most significant bits (MSB) first in two and four bit transfer modes. This allows for the data to be received by the host computer, minimally coalesced and then left shifted into a variable. Each multiplexer (mux) has a singular function in the inventive design. That function is the routing of a given signal (or group of signals) to the appropriate pin(s). Mux #1 selects between upper and lower nibbles and is used in two and four bit modes. Mux #2 selects between the upper pair and lower pair during two bit mode only. In four bit

mode, Mux #2 passes Mux #1's lower two bits straight through to a subsequent mux. Mux #3 controls the status lines whenever they are necessary for data transfer, which is in all slave modes. In two bit mode, the AUTOFEED, PE, BUSY and ACK signals are used. In four bit mode, the AUTOFEED, ERROR_OUT, SLCT_OUT, PE, ACK and BUSY signals are used. These signals travel through mux #5. In eight bit mode, the only control lines used are AUTOFEED and ACK. Mux #4 controls the AUTOFEED line during DMA/MASTER mode so that a computer interface 520 may communicate with another computer interface device 520 as if it was a host computer Master.

Figures 21-23 illustrate the relationship between the above-described registers and the host interface signals generated in each of the three Data Director 545 modes of operation: 2-bit data transfer mode, 4-bit mode, and 8-bit mode. As shown in Figure 21 for two bit mode, the AUTOFEED, PE, BUSY and ACK signals are used. As shown in Figure 22 for four bit mode, the AUTOFEED, ERROR_OUT, SLCT_OUT, PE, ACK and BUSY signals are used. As shown in Figure 23 for eight bit mode, the only control lines used are AUTOFEED and ACK.

Figure 24 is a table illustrating the relationship between the operating modes of Data Director 545 (i.e. 2-bit, 4-bit, 8-bit, slave, and master mode) and the state of the control and data lines provided to host interface 522 by the computer interface device 520 of the present invention.

Having selectively programmed the hardware registers as described above, the EPC 533 and the Data Director 545 of control logic 532 use the register control selections to appropriately strobe data across the host interface 522. In slave mode, the computer interface device 520 responds to the AUTOFEED signal provided by the host computer across host interface 522. This response comprises a handshaking protocol using the AUTOFEED signal (driven by the host computer 510 in slave mode) and the ACKNOWLEDGE signal (driven by the computer interface device

520 in slave mode). The handshaking protocol is used to move data at two, four, or eight bits at a time across the host interface 522.

Data transfers between the host computer 510 and the computer interface device 520 can be initiated in two ways: Interrupt Transfer Mode and Polled Transfer Mode. Host computer 510 can be interrupted by a change in the state of the ACK signal line. This may occur if the computer interface device 520 sets this signal line to indicate that it wants to begin a selection sequence. This type of data transfer initiation is called Interrupt Transfer Mode. In this mode, transfers can be initiated by the computer interface device 520, which examines the data lines and the AUTOFEED line to establish that the host computer 510 has not yet begun a selection sequence. If this is the case, the computer interface device 520 sets the ACK line and waits for an address to be presented on the data lines. The host computer 510 begins the selection sequence by changing the data lines to the address of the computer interface device 520. When this occurs, the computer interface device 520 clears the ACK line and continues the selection sequence described in more detail below.

If the host computer 510 is not configured to be interrupted by ACK, transfers must be initiated by the host computer 510 in a Polled Transfer Mode. In this mode, the host computer 510 executes the selection sequence approximately every 50 milliseconds to see if the computer interface device 520 has data that it wishes to transfer to the host computer 510. Whether initiated by an ACK interrupt, or a polling event, computer interface device 520 to host computer 510 communication always follows the successful completion of a selection sequence, which is described in more detail in the following paragraphs.

Because the external port connecting the host computer 510 to the computer interface device 520 can be shared with other devices, a specific handshake sequence is required for a communication session to begin. As shown in Figure 27, this sequence is designed to minimize the possibility of interference between the computer

interface device 520 and other devices connected to the same external port lines.

Referring to Figure 27, the selection sequence consists of the following steps: The host computer 510 outputs a unique address on data lines 0:7 at time 3010 and changes the AUTOFEED signal from a high to a low state at time 3012. Computer interface devices that have just been powered up or reset will respond if data lines 0:7 have the value 7, while an initialized computer interface device 520 will respond to an address that was passed to it during initialization (typically 0x80 to 0x84 in the preferred embodiment). At time 3016, Switch Logic 524 is set so that the ACKNOWLEDGE signal line to the peripheral interface 528 is disconnected. At time 3014, the Control Logic ACKNOWLEDGE signal line is enabled as an output signal by setting the ACKNOWLEDGE signal high. The ACKNOWLEDGE signal is kept high for a minimum of 50 microseconds in the preferred embodiment. At time 3018, the ACKNOWLEDGE signal is set low. In response, the host computer 510 sets the AUTOFEED signal high at time 3020. In response to the AUTOFEED signal, computer interface device 520 sets the ACK signal high at time 3022. The handshaking sequence continues between the AUTOFEED and ACK signals for an additional three cycles. Specifically, the computer interface device 520 waits for the AUTOFEED signal line to go low. Then, computer interface device 520 sets the ACKNOWLEDGE signal low and again waits for the host computer to set the AUTOFEED signal high. In response to this, the computer interface device 520 sets the ACKNOWLEDGE signal high. Following this sequence, the Peripheral Interface Disable line 426 is set, so that all lines in signal group 4200 are disconnected. A failure to complete any of the steps in the selection sequence within 64 milliseconds causes the computer interface device 520 to return to the idle state and to wait for the first step of the selection sequence to begin. Once the selection sequence is successfully completed, data transfers consist of a series of data packets. Each data packet is preceded by a one

byte command sent by the host computer 510. Depending upon the value of this initial command byte, subsequent handshakes can transfer data from the host computer 510 on data lines 0:7, or to the host computer 510 in 2, 4, or 8 bits per handshake, or in both directions simultaneously.

The present invention also provides a procedure for initial program loading (IPL or Bootload) of the computer interface device 520. In order to minimize the amount of ROM 542 storage on the computer interface device 520 and to allow for updates to the programs that are run on the computer interface device 520, only a small IPL program is stored in ROM 542 on the computer interface device 520. The function of this IPL program is to wait for a selection sequence wherein the address presented on the data lines is address 7. In this case, the IPL program then receives an object program from the host computer 510. This process begins after reset or power-up. At reset or power-up, CPU 538 of computer interface device 520 sets the control logic 532 so that all signal lines are inputs. Similarly, CPU 538 sets switch logic 524 to connect all signal lines from the host computer 510 to the peripheral interface 528. The CPU 538 initiates communication with the host computer 510 by first setting the INTERFACE ENABLE control line and then waiting until the selection sequence described above has been completed. Next, the host computer 510 sends a command packet to the computer interface device 520 in order to obtain the hardware configuration of the computer interface device 520. Finally, the host computer 510 sends a command to the computer interface device 520 which contains the object program. The object program is transferred to RAM 544 and stored. The IPL program transfers control to the newly downloaded object program in RAM 544 once the object program is successfully loaded.

The present invention provides a unique handshaking protocol that includes a means for reversing the direction of a data transfer with no transmission delay or the need to re-cycle the control signals used to throttle the transfer. This protocol also provides for

simultaneous bi-directional data transfer. Referring now to Figures 25 and 26, timing diagrams illustrate this handshaking protocol. In Figure 25, a unidirectional data transfer sequence (in slave mode) from the host computer to the computer interface device 520 is illustrated. In this unidirectional case, data from host computer 510 is presented on data bus lines 0:7 (i.e. first data lines) of the host interface 522 at time 2505. Next, the host computer 510 asserts the AUTOFEED signal at time 2507 to indicate that data is valid on the data bus. AUTOFEED is only asserted when data is valid. In response, the EPC 533 presents a status word on the 2, 4, or 8 bit lines for transferring data to host computer 510 (i.e. second data lines) at time 2510. EPC 533 then reads the data from the data bus and asserts the ACKNOWLEDGE signal at time 2511 to indicate that the data is latched. In response to the asserted ACKNOWLEDGE signal, the host computer 510 reads the status word and deasserts the AUTOFEED signal at time 2512 to acknowledge reading the status and to acknowledge the latching of the data by the computer interface device 520. Finally, the ACKNOWLEDGE signal is deasserted in response to the deasserted AUTOFEED signal at time 2514. Starting at time 2516, a new data transfer cycle using the same protocol can be initiated for the next data item to be sent by the host computer 510 to the computer interface device 520.

Referring now to Figure 26, a bi-directional data transfer sequence (in slave mode) between the host computer and the computer interface device 520 is illustrated. In the bi-directional case, the host computer first sends a read request to the computer interface device 520. In response to this request, the computer interface device 520 sends one or more data items (2, 4, or 8 bits wide) back to the host computer 510.

The bi-directional sequence starts when a read request from host computer 510 is presented on data bus lines 0:7 of the host interface 522 at time 2605. Next, the host computer 510 asserts the AUTOFEED signal at time 2607 to indicate that data is valid on the data bus. Next, the EPC 533 presents a status word on the 2 or 4 bit

lines for transferring data to host computer 510 at time 2610. The EPC 533 reads the read request from the data bus and asserts the ACKNOWLEDGE signal at time 2611 to indicate that the read request is latched. In response to the asserted ACKNOWLEDGE signal, the host computer 510 deasserts the AUTOFEED signal at time 2612 to acknowledge the latching of the read request by the computer interface device 520. Next, the ACKNOWLEDGE signal is deasserted in response to the deasserted AUTOFEED signal at time 2614. At this point in the sequence, the EPC 533 decodes the read request.

Starting at time 2616, the host computer 510 again asserts the AUTOFEED signal to indicate that a reversal of the data transfer direction has occurred. Now, the host computer is ready to receive data from the EPC 533, or optionally, to send the next byte of data to EPC 533. In response to the data transfer direction reversal, EPC 533 presents the first byte (or 2 or 4 bit nibble) of the requested data on the 2, 4, or 8 bit data lines at time 2618. Then, EPC 533 asserts the ACKNOWLEDGE signal at time 2619 to indicate that valid data is present on the data lines. The host computer 510 reads the data on the data lines and deasserts the AUTOFEED signal at time 2620 to indicate that the data has been latched. In response to the deasserted AUTOFEED signal, the EPC 533 may invalidate the data at time 2622 and then deassert the ACKNOWLEDGE signal at time 2623. The AUTOFEED signal is asserted at time 2624 to set up for the next data item transfer from the EPC 533 to the host computer 510. In response to the assertion of the AUTOFEED signal, the EPC 533 presents the next data item on the data lines at time 2626 and asserts the ACKNOWLEDGE signal at time 2627 to indicate to the host computer 510 that valid data is present on the data lines. In a series of repeated cycles, the host computer reads and acknowledges subsequent data items sent by computer interface device 520 to the host computer 510. As evident, the invented data transfer protocol reverses the data transfer direction and the sense of the control signals without the loss of a handshake.

As an optional extension to the protocol illustrated in Figure 26, the host computer 510 may initiate the transfer of data items 2642 and 2644 from the host computer 510 to the computer interface device 520 at nearly the same time that data items 2646 and 2634 are transferred from the computer interface device 520 to the host computer 510. This near-simultaneous bi-directional transfer of data is accomplished using the rising and falling edges of the AUTOFEED signal for triggering the bi-directional transfer of data. Prior to asserting the AUTOFEED signal at time 2616, the host computer 510 presents datum 2642 on data bus lines 0:7 of the host interface 522. The host computer 510 then asserts the AUTOFEED signal at time 2616. In response to the AUTOFEED signal, the computer interface device 520 performs two functions, 1) computer interface device 520 presents datum 2646 on the 2, 4, or 8 bit data lines, and 2) computer interface device 520 reads datum 2642 from data lines 0:7. The computer interface device 520 acknowledges the completion of both tasks by asserting the ACKNOWLEDGE signal at time 2619. The host computer 510 reads the data on the data lines and deasserts the AUTOFEED signal at time 2620 to indicate that the data has been latched. The host computer 510 may also remove datum 2642 from data lines 0:7 at this time. A similar near-simultaneous bi-directional data transfer starts at time 2624 for data items 2644 and 2634. Thus, the protocol of the present invention may be used to transfer data in two directions at nearly the same time.

Referring now to Figures 28-33, the processing logic used in the present invention is illustrated. These figures and the following detailed description of the preferred embodiment describe the processing logic used with the computer interface device of the present invention. A portion of the processing logic is resident in the host computer. Another portion of the processing logic is resident in the computer interface device. A detailed description of both portions of this processing logic in support of the present invention follows.

Peer-to-peer networking for personal computers (PC's) has previously been implemented by loading a large networking program into the limited memory of the PC, which remains there and reduces the amount of free memory available for other applications programs to run. A major advantage of the hardware architecture of the present invention is that much of the networking software resides on the computer interface device 520 of the present invention and is executed therefrom by CPU 538. This design frees up much of the host computer 510 memory space and allows the host computer 510 to execute local application programs simultaneously. The following sections will outline this software partitioning.

In a typical peer-to-peer network, each host computer performs a dual function: (1) the host computer can be a file or print client of remote host computers and printers, and/or (2) the host computer can act as a file server for remote clients. In the present invention, the host computer is not burdened with the print server functionality, which resides entirely within the computer interface device 520. In addition, the computer interface device 520 supports host computer access to additional services such as direct access to network protocols, modem dial-in or dial-out, or processing of electronic mail.

Referring now to Figure 28, a block diagram of the processing logic of the present invention is illustrated. A portion 2808 of this processing logic is stored in random access memory 544 and executed therefrom by CPU 538. Another portion 2806 of this software is stored in a memory of host computer 510 and executed therefrom by the host computer 510.

To implement the client file and print functionality, the host computer portion of the processing logic 2806 includes a file system and print service module 2810, which is added to the basic host computer software. This module 2810 intercepts file service requests and print service requests that may need to be redirected over the network. These requests are converted into one or more data packets that are transferred to the computer interface device

520 using the Host External Port Session Manager (HEPSM) service 2812. The HEPSM formats packets of information corresponding to the request and delivers the packets of information to the computer interface device 520 via Host External Port Link Manager (HEPLM) service 2814. The information packets are retransmitted if errors occur during transfer. The HEPSM then waits for a reply from the computer interface device 520, which is returned to the host computer 510 client program.

Such a series of operations is commonly referred to as a "remote procedure call", where the execution of the requested client operation is achieved by transferring all information associated with the operation from the host computer 510 through HEPSM/HEPLM to the computer interface device 520. The requested operation is then executed by software resident on the computer interface device 520, and the results are returned to the host computer 510. The host computer 510 then resumes execution as if the procedure had been executed and completed locally.

Once service requests are transferred via host computer 510 HEPSM/HEPLM services to the computer interface device 520, a corresponding External Port Link Manager (EPLM) service 2816 and a External Port Session Manager (EPSM) service 2818 receives the request. Client print requests are processed by a print manager service 2820 in the computer interface device 520. This print manager 2820 sends print data for a local printer through the switch logic section 524 described above. Alternatively, the print manager 2820 executes an AppleTalk PAP (Printer Access Protocol) network transaction using PAP module 2824 when the destination of the print request is a remote printer. The AppleTalk PAP protocol was chosen in the preferred embodiment so that the large number of Postscript™ laser printers manufactured by Apple™ Computer, Inc. and other vendors would be directly accessible by the computer interface device 520.

Client file transactions are handled by the File Manager service 2822 in the computer interface device 520. This service automatically establishes the optimal network file protocol to use in

communicating with the required file server. In a typical case where a host computer 510 is making a request of a compatible file server, the Direct File Protocol (DFP) 2828 is used. In this case, the compatibility between the client and the server eliminates the need to translate a request between client and host file system semantics. In other cases, the File Manager 2822 employs a protocol translation service in the computer interface device 520 to convert the host client's file system request into the network filing protocols implemented by the file server. One example of this translation is the AFP (AppleTalk Filing Protocol) translation service 2826, which converts IBM™ PC client file requests into an Apple Macintosh server compatible form. Other examples of this translation are shown as additional network file protocols 2830 in Figure 28. It will be apparent to those of ordinary skill in the art that conventional methods exist for converting from one network protocol to another. The present invention, however selects from several available translation modules depending upon the protocol corresponding to the client and the server of a particular service request.

The computer interface device 520 supports simultaneous multiple connections to multiple server types and multiple service types. Multiple server types are supported if, for example, the host computer 510 makes a request of a compatible file server through DFP 2828 followed by a request to an incompatible file server through a translation service such as AFP 2826. Multiple service types are supported if, for example, the host computer 510 makes a request for network service through file manager 2822 followed by a request for printer service through print manager 2820. In each case, multiple connections to multiple server types and multiple service types may be pending at the same time. In addition, the computer interface device 520 maintains status of the pendency of requests and status of the availability of services. Thus, the computer interface device 520 can quickly determine if a particular service has gone down or is experiencing errors.

Executing AFP transactions on the computer interface device 520 requires that a whole set of AppleTalk protocols be implemented in the computer interface device 520. This set of protocols includes LLAP, DDP, NBP, ZIP, ATP, and RTMP. These protocols are well known to those of ordinary skill in the art. Although DFP 2828 currently uses the same protocol set as AFP 2826 in the preferred embodiment, other protocol sets can easily be employed in an alternative embodiment (e.g. IPX/SPX, developed by Netware™, Inc., LAN Manager™, developed by Microsoft™, Inc., or TCP/IP/NFS). The ability of the present invention to simultaneously support multiple heterogeneous network servers in the same system is one of the unique benefits of the system architecture of the present invention.

File server functionality similarly benefits from this system architecture. File or print service requests are received by the computer interface device 520 via a network port. These requests are transferred to a compatible protocol processing module. For IBM™ PC and DOS compatible file clients sending requests to an IBM™ PC and DOS compatible file server, the DFP 2828 handles the request. The DFP 2828 transfers the request to the host computer 510 server via EPSM/EPLM transactions. These requests are directly executed by DOS on the host computer 510 server.

For non-DOS clients, the File Manager 2822 on the computer interface device 520 provides translation services that convert the client's file system requests into a set of DOS transactions for a DOS compatible server. For example, when the network file client is an Apple Macintosh™ computer, the AFP requests received from the client are converted by the AFP server 2826 in the computer interface device 520 into a series of DOS file operations. This translation process almost always results in reduced compatibility and frequently limits the network file client's file operations; because, the DOS file system does not support all of the file system capabilities present in the native file system of some clients. Nevertheless, the ability to provide network file services across a

wide range of file systems and network protocols is unique and highly desirable.

The present invention provides the advantage of isolating all but the highest level of access to network and peripheral interface communications control functions within the computer interface device 520. In this manner, these control functions are off-loaded from the host computer 510. This off-loading of the host computer 510 saves memory storage in the host computer. In addition, off-loading also allows the computer interface device 520 to handle varied types of host computers. Because the interface between the host computer and the computer interface device operates at a higher functional level than provided in prior art systems, the computer interface device is able to support various types of host computer platforms with little or no modification to the hardware and software of the computer interface device.

Finally, the architecture of the present invention is not limited to the parallel port and LocalTalk physical media. An ETHERNET interface 2840 may be added to the network ports of the computer interface device 520. In an alternative embodiment, host interface 116 comprises a serial port connection for coupling the computer interface device 110 with host computer 112 across a serial interface. In yet another alternative embodiment, host interface 116 comprises an infrared port connection for coupling the computer interface device 110 with host computer 112 across a wireless infrared data link. Other conventional data communications methods may equivalently be used. In general, parallel, serial, and infrared data communications methods are well known to those of ordinary skill in the art.

Referring now to Figures 29-33, flowcharts are used to illustrate the processing logic used by the present invention. Referring now to Figure 29, the processing logic for handling a host client file or print request is illustrated. Such a request is originated by a host computer acting as client. The file or print request is issued to the file system and print services module 2810 by other software executing within the host client (processing block

2912). If the request is for local host file service, processing path 2916 is taken to processing block 2920 where the local file service request is routed to a local host file system existing in conventional computer systems. Processing then terminates through the bubble labeled B at termination bubble 3036 illustrated in Figure 30.

Referring again to Figure 29, if the request is for print service or remote file system service, processing path 2918 is taken to processing block 2922. The request is prepared for routing to the computer interface device 520 by converting the request to a set of external port session manager (EPSM) compatible data packets. These packets are then sent to the host external port session manager (HEPSM) 2812 in processing block 2922. The HEPSM 2812 formats the data packets for transfer to the computer interface 520 via host external port link manager (HEPLM) 2814 (processing block 2924). The HEPLM 2814 then transfers the request to the computer interface device 520 via an external port on the host computer 510. An external port link manager (EPLM) 2816 in computer interface device 520 receives the request and transfers the request to the external port session manager (EPSM) 2818 in processing block 2926. Control then transfers to the bubble labeled A as illustrated in Figure 30.

Referring now to Figure 30, the processing logic for a host client transaction continues. At the bubble labeled A, if the request received by the EPSM 2818 of computer interface device 520 is not a request for printer service, processing path 3011 is taken to the bubble labeled C illustrated in Figure 31 where a remote file service request is processed. If the request is for printer service, processing path 3013 is taken to processing block 3018. Print service requests are processed by print manager 2820. The print request is transferred by print manager 2820 to printer access protocol (PAP) module 2824 in processing block 3018. Next, a protocol module compatible with the print request and corresponding printer is selected in processing block 3020. For example, if an AppleTalk computer network is used, an AppleTalk printer access protocol is used to format the print request for

transfer across the network. The request is formatted using the selected protocol in processing block 3022. The formatted request is sent across the network via a network port driver for remote print requests or to a local printer driver for local print requests in processing block 3024. If a local printer is selected, standard processing logic routes the print request to the selected printer. Status is received from the selected printer driver in processing block 3026. This status is sent back to the host client computer through print manager 2820, EPSM 2818, EPLM 2816, HEPLM 2814, HEPSM 2812, and file system and print services module 2810 in processing block 3034. Processing for the host client print request then terminates at termination bubble 3036 illustrated in Figure 30.

Referring now to Figure 31, the processing logic for a host client remote file service request is illustrated. These requests are received by file manager 2822 from EPSM 2818. If the remote file service request does not require translation to a different protocol, processing path 3114 is taken to processing block 3116 where the request is routed from a file manager 2822 to direct file protocol (DFP) 2828. The DFP module 2828 is used if the client computer is compatible with the server.

If, however, the client is not compatible with the server, the file service request must be translated from one protocol to another. In this case, processing path 3112 is taken to processing block 3118 where an appropriate translation service is selected. For example, the AppleTalk file protocol (AFP) 2826 converts IBM PC and DOS compatible file requests into transactions compatible with an Apple Macintosh file server. Having selected the appropriate translation service, the request is translated in processing block 3120. A compatible network protocol module is selected in processing block 3122. The request is sent to the selected network protocol driver and the corresponding network media driver in processing block 3124. The client request is thereby sent across the network to a remote file server. Status is received from the remote file server in processing block 3126. This status is sent back to the host client

through file manager 2822, EPSM 2818, EPLM 2816, HEPLM 2814, HEPSM 2812, and file system and print services module 2810 (processing block 3128). Processing then terminates for the host client request through the bubble labeled B illustrated in Figure 30.

Referring now to Figure 32, host server processing logic is illustrated for processing requests received by a host server from a network. In this case, a network media driver receives a print or file request via a network port in processing block 3210. A network protocol module compatible with the remote client is selected in processing block 3212. The request is routed to the selected network protocol in processing block 3214. If the incoming request is for local print service, processing path 3218 is taken to processing block 3222. In this case, the print request is routed to print manager 2820 for printing through printer driver 2834. Note that for a local print request received from a remote client, the host server computer is not involved in the transaction. The entire transaction is handled by computer interface device 520. Once the print request has been routed through printer driver 2834, the status corresponding to the print request is received by print manager 2820 in processing block 3224. This status is routed back to the remote print client by print manager 2820. Control then transfers to the bubble labeled E illustrated in Figure 33 where the status is sent back to the remote client.

Referring again to Figure 32, if the request is not for local print service, processing path 3220 is taken to decision block 3226. At this point, the remote client protocol is compared with host server protocol. If the file service request requires translation from one protocol to another, processing path 3230 is taken to processing block 3234 where a corresponding translation service is selected. The request is routed to the selected translation service and the request is translated into a protocol corresponding to the host server in processing block 3236. Processing then continues at the bubble labeled D as illustrated in Figure 33.

Referring again to decision block 3226, if the file service request does not require translation to a different protocol, processing path 3228 is taken to the block labeled 3232 where the request is routed to the direct file protocol (DFP) module 2828. The DFP module 2828 transfers the untranslated request to file manager 2822 where the request is sent directly to the host file server. Processing then continues at the bubble labeled D illustrated in Figure 33.

Referring now to Figure 33, processing for the remote file service request is illustrated. In processing block 3310, the file service request is routed to EPSM 2818 and EPLM 2816 for transfer to host computer 510. The host file server receives the request via the HEPLM 2814 and HEPSM 2812. The host server performs the requested service in processing block 3212. The results or status of the request performed by the host server is routed back to the file manager 2822 of computer interface device 520 via HEPSM 2812, HEPLM 2814, EPLM 2816, and EPSM 2818 (processing block 3214). If translation of the information from one network protocol to another is required, processing path 3220 is taken to processing block 3222 where the results or status of the requested operation is translated into a protocol compatible with the client. If, however, no translation is required, processing path 3218 is taken to processing block 3224. The results of the requested operation are sent to the appropriate network protocol driver in processing block 3224. Processing for the host server software of the present invention then terminates at end bubble 3226.

Referring again to Figure 28, software within the computer interface device 520 further includes a Name Table 2852. The Name Table 2852 is coupled to File Manager 2822 and EPSM 2818. Name Table 2852 is used to manage the names of network clients and servers and other network entities coupled to the computer interface device 520. The Name Table 2852 can be used to associate network entities in a file system context or for purposes not related to a file system. This table is managed by the computer

interface device 520 itself. The Name Table 2852 includes storage for a network client or server name and a host compatible name for each network client and server (i.e. node) along with processing logic for accessing and manipulating data in the Name Table 2852. Because the naming conventions for various network clients and servers and corresponding files may be different, Name Table 2852 software is included for resolving name conflicts between network nodes (i.e. clients and servers). For example, AppleTalk compatible nodes can have up to 32 character names. DOS compatible nodes, on the other hand, can have only eight character names with a three character extension. Name Table 2852 software maps longer network node names into shorter host compatible names, or vice versa, using the Name Table 2852. Because this mapping may cause previously unique names to become equal and thereby no longer unique, the Name Table 2852 maintains the original name and its sort order to maintain uniqueness between the node and file names. By using the original sort order, the present invention allows the translated (mapped) names to appear in the same order (and mapping) on all nodes without a central names server. In this manner, the Name Table 2852 can be used to maintain a list of network nodes, which can be queried and used by a host computer 510.

The content of Name Table 2852 is dynamically maintained by the computer interface device 520 without the need for host computer 510 intervention. Computer interface device 520 monitors network transactions between network nodes during normal operations. The node names used in each transaction can be identified. These transaction node names are compared by Name Table 2852 software with the current contents of the Name Table 2852. If the transaction node name is not found in the Name Table 2852, the name is saved in the Name Table 2852 as a potentially new node name (i.e. an unconfirmed new node name). The computer interface device 520 confirms that the potentially new node name is, in fact, a valid node name by either attempting a service connection to the new node, or by receiving an explicit

network name confirmation via another monitored network transaction. Names are removed from the Name Table 2850 using a similar process. By monitoring network node-to-node transactions, the computer interface device 520 determines when a node service failure, a request confirmation failure, or a lack of traffic for a particular node has occurred. In this case, the identity of the unresponsive node is removed from the Name Table 2850. In addition, a name can be explicitly removed from the Name Table 2850 upon request from host computer 510 or upon request from the computer interface device 520 of another node. In this manner, the computer interface device 520 dynamically maintains an accurate list of network nodes in the Name Table 2850. A host computer or other node may at any time query the content of the Name Table 2850 stored in a particular computer interface device 520.

The present invention includes an efficient way of querying the content of the Name Table 2850 and thereby querying the identity of all file servers present on a particular network. This method is derived from the resource naming convention used in DOS compatible systems for identifying disk drives (i.e. A:, B:, or C:) in a particular computer system. The present invention uses the identifier N: to represent a directory of file servers or nodes present on a particular network. This directory corresponds to the names maintained in Name Table 2850. Each node name or node user name is represented as a sub-directory of the N: directory. For example, N:\John_Smith represents a computer on the network used by or identified by the name John Smith. It will be apparent to those of ordinary skill in the art that other equivalent identifiers of the directory of file servers may be used. For example, X: or any_string: could be used in a DOS based system or \net\ or \any_network_string\ could be used in a UNIX based system. In any case, a predefined identifier is established to represent the network directory of file servers or nodes of a particular network.

The processing logic of the present invention includes means for receiving and processing a network directory identifier used in a

host computer 510 query of network nodes. Referring again to Figure 28, this network directory identifier is received by file system and print services module 2810. Included therein is logic for recognizing the predefined network directory identifier and the request for a query of network nodes. In response, file system and print services module 2810 generates a request to file manager 2822 of computer interface device 520 for the contents of Name Table 2852. The contents of Name Table 2852 comprises the dynamically updated list of network nodes. This list is transferred back to file system and print services module 2810, formatted for display to a user of host computer 510, and subsequently displayed as a list of network nodes. Depending upon the type of host computer (i.e. DOS or Apple compatible), the list is provided in a compatible form.

The processing logic of the present invention also includes means for receiving and processing a network directory identifier used in a query of files residing in a particular network node. This logic also includes means for receiving and processing a network directory identifier used in a file transfer request across the network. Files in the network of the present invention exist in one of two modes: shared or private. Shared files are visible and accessible to other nodes on the network. Private files are visible and accessible only to an owner node. Shared files may be read or written by any node on the network. Private files may be written from one node to another; but, private files may not be read by any node other than the owner node (i.e., the node in which the private file resides).

These two file modes are implemented in the present invention by maintaining two sub-directories in the hard disk or other permanent storage medium of each node. The first subdirectory is used for storage of network shared files; the second sub-directory is used for storage of private files. In the preferred embodiment, the private file sub-directory is itself a sub-directory of the shared file sub-directory. It will be apparent to those of ordinary skill in the art that the private sub-directory may equivalently be

positioned at the same or a different hierarchical level relative to the shared file directory. For example, a shared file sub-directory for Node_X (node identifier) having a disk resource designated C: is identified by the string C:\shared. A private file sub-directory for Node_X having a disk resource designated C: is identified by the string C:\shared\private. It will be apparent to those of ordinary skill in the art that other equivalent identifiers of the shared and private file directories may be used.

Having established shared and private file directories for each node, other network nodes may then transfer shared or private files to Node_X (or equivalently to any other node) by transferring files to the shared or private directories of Node_X using the network directory identifier. For example, a shared file may be sent from any network node to Node_X using the destination directory identifier string N:\Node_X\shared. Similarly, a private file may be sent from any network node to Node_X using the destination directory identifier string N:\Node_X\shared\private. File Manager 2822 includes logic for detecting a file transfer request to or from a shared or private directory of a particular node. Upon decoding the network directory identifier and the node identifier of the host computer, the File Manager 2822 routes an incoming file to the C:\shared or C:\shared\private directory on the host computer 510 depending upon whether the shared or private directory designation was used in the file transfer request. The file is so routed by stripping the network directory identifier and the node identifier and inserting an identifier of the permanent storage medium of the host computer on which the shared and private file directories are stored (i.e. C: in the above example). This modified file transfer request is sent to the file system and print services module 2810 and then on to the file system software of the host computer for conventional processing. For private file transfers, the file access permission of the file is set to prevent other network nodes from reading, deleting, executing, or otherwise knowing of the existence of the private file. In this manner, the present invention implements a network file transfer protocol that is

substantially invisible to the host computer 510 to which the computer interface device 520 is connected.

Thus, a computer interface device providing a networking and peripheral interface capability using an external port of a host computer is described. Although the present invention is described herein with reference to a specific embodiment, many modifications and variations therein will readily occur to those of ordinary skill in the art. Accordingly, all such variations and modifications are included within the intended scope of the present invention as defined by the following claims.

CLAIMS

What is claimed is:

1. A computer interface device comprising:

a host interface for connecting said computer interface device with a host computer, said host interface having a plurality of host data lines and a plurality of host control lines;

a peripheral interface for connecting said computer interface device with a peripheral device, said peripheral interface having a plurality of peripheral data lines and a plurality of peripheral control lines, said host computer using a first portion of said host control lines and a first portion of said peripheral control lines for controlling communication with said peripheral device; and

a control unit coupled to said host interface and said peripheral interface, said host computer using a second portion of said host control lines for controlling communication with said control unit, said first portion of said host control lines being different from said second portion of said host control lines, said control unit selectively coupling or uncoupling said host data lines and said host control lines with said peripheral data lines and said peripheral control lines.

2. The computer interface device as claimed in claim 1 wherein said control unit includes an analog switch for selectively coupling or uncoupling said host data lines and said host control lines with said peripheral data lines and said peripheral control lines.

3. The computer interface device as claimed in claim 1 wherein said host interface and said peripheral interface is an eight bit parallel interface.

4. The computer interface device as claimed in claim 1 wherein said host interface and said peripheral interface is a serial interface.

5. The computer interface device as claimed in claim 1 wherein said host interface is a parallel interface and said peripheral

interface is a serial interface, said computer interface device further including signal conversion logic coupled to said host interface and said peripheral interface for converting signals received on said host interface to signals compatible with said peripheral interface and for converting signals received on said peripheral interface to signals compatible with said host interface.

6. The computer interface device as claimed in claim 1 wherein said control unit includes a processor (CPU) and a random access memory.

7. The computer interface device as claimed in claim 1 wherein said second portion of said host control lines comprises (a) an AUTOFEED signal used by said host computer to strobe data on said host data lines, and (b) an ACKNOWLEDGE signal used by said computer interface device to acknowledge a transfer of data on said host data lines.

8. The computer interface device as claimed in claim 1 wherein said control unit includes a programmable configuration register, said register including an indication of one of a plurality of host computer types to which said computer interface may be coupled, said control unit including control logic for interfacing with a host computer of each of said plurality of host computer types.

9. In a computer interface device having a host interface for connecting said computer interface device with a host computer, said host interface having a plurality of first data lines and a plurality of second data lines, a first control line, and a second control line, a process for controlling data transmission between said host computer and said computer interface device, said process comprising the steps of:

presenting a read request on said first data lines, said read request being presented by said host computer;

asserting said first control line to indicate the presence of valid data on said first data lines;

latching said read request from said first data lines, said read request being latched by said computer interface device;

asserting said second control line to indicate the completion of said step of latching said read request;

deasserting said first control line to acknowledge assertion of said second control line;

deasserting said second control line to acknowledge deassertion of said first control line;

asserting said first control line to indicate the availability of said host computer to receive data on said second data lines;

presenting data on said second data lines, said data being presented by said computer interface device;

asserting said second control line to indicate the presence of valid data on said second data lines;

latching said data from said second data lines, said data being latched by said host computer;

deasserting said first control line to indicate the completion of said step of latching said data; and

deasserting said second control line to acknowledge deassertion of said first control line.

10. The process as claimed in claim 9 wherein said host interface is a parallel interface.

11. The process as claimed in claim 9 wherein said first control line is an AUTOFEED signal and said second control signal is an ACKNOWLEDGE signal.

12. The process as claimed in claim 9 further including the steps of:

presenting data on said first data lines, said data being presented by said host computer;

asserting said first control line to indicate the presence of valid data on said first data lines;

latching said data from said first data lines, said data being latched by said computer interface device in response to said asserting said first control line step;

presenting data on said second data lines, said data being presented by said computer interface device in response to said asserting said first control line step;

asserting said second control line to indicate the presence of valid data on said second data lines; and

latching said data from said second data lines, said data being latched by said host computer in response to said asserting said second control line step.

13. A computer interface device comprising:

a host interface for connecting said computer interface device with a host computer, said host interface having a plurality of host data lines and a plurality of host control lines;

a peripheral interface for connecting said computer interface device with a peripheral device, said peripheral interface having a plurality of peripheral data lines and a plurality of peripheral control lines; and

a control unit coupled to said host interface and said peripheral interface, said control unit including means for selectively sending a two bit, a four bit, or an eight bit data item to said host computer, said two bit and said four bit data items being sent to said host computer on said host control lines.

14. The computer interface device as claimed in claim 13 wherein said two bit data item is sent on a paper empty signal line and a busy signal line of said host control lines.

15. The computer interface device as claimed in claim 13 wherein said four bit data item is sent on a paper empty signal line, a busy

signal line, a select signal line and an error signal line of said host control lines.

16. A computer interface device comprising:

a host interface for linking said computer interface device with a host computer, said host interface providing a plurality of data signals and a plurality of control signals;

a peripheral interface for linking said computer interface device with a peripheral device, said peripheral interface providing a plurality of data signals and a plurality of control signals, said host computer using a first portion of said control signals for controlling communication with said peripheral device; and

a control unit coupled to said host interface and said peripheral interface, said host computer using a second portion of said control signals for controlling communication with said control unit, said first portion of said control signals being different from said second portion of said control signals, said control unit selectively coupling or uncoupling said host interface with said peripheral interface.

17. The computer interface device as claimed in claim 16 wherein said host interface and said peripheral interface is an infrared data link.

18. A computer interface device comprising:

a host interface for connecting said computer interface device with a host computer, said host interface having a plurality of data lines and a plurality of control lines;

a network interface for connecting said computer interface device with a network, said network having a plurality of network nodes attached thereto; and

a file manager coupled to said network interface and said host interface for controlling file transfers between said network and said host computer, said file manager including logic for decoding network query having a network directory identifier, said file

manager responding with a list of network nodes currently attached to said network.

19. The computer interface device as claimed in claim 18 further including a name table coupled to said file manager, said name table retaining an identifier of each network node on said network.

20. The computer interface device as claimed in claim 18 wherein said file manager further including logic for decoding a file transfer request between said host computer and one of said plurality of network nodes, said file transfer request having a network directory identifier, said logic for decoding a file transfer request further including logic for converting said network directory identifier to an identifier corresponding to a storage resource of said host computer on which files are stored.

21. The computer interface device as claimed in claim 18 wherein said file transfer request also having an indication of whether a shared or private file is being transferred.

22. The computer interface device as claimed in claim 18 further including a network idle timer for maintaining a running count of a length of time that said network has been idle, said network idle timer being reset when information transfer activity is present on said network.

23. The computer interface device as claimed in claim 18 further including a host computer resident software module and computer interface device resident software for translating host computer specific print and network service requests to a form compatible with a destination device, said host computer resident software module containing information specific to a particular type of host computer, said computer interface device resident software containing information for translating said service requests for a plurality of different types of host computers.

24. A computer network comprising:

a computer interface device including:

a.) a host interface for connecting said computer interface device with a host computer, said host interface having a plurality of host data lines and a plurality of host control lines;

b.) a peripheral interface for connecting said computer interface device with a peripheral device, said peripheral interface having a plurality of peripheral data lines and a plurality of peripheral control lines; and

c.) a network interface for connecting said computer interface device with a network; and

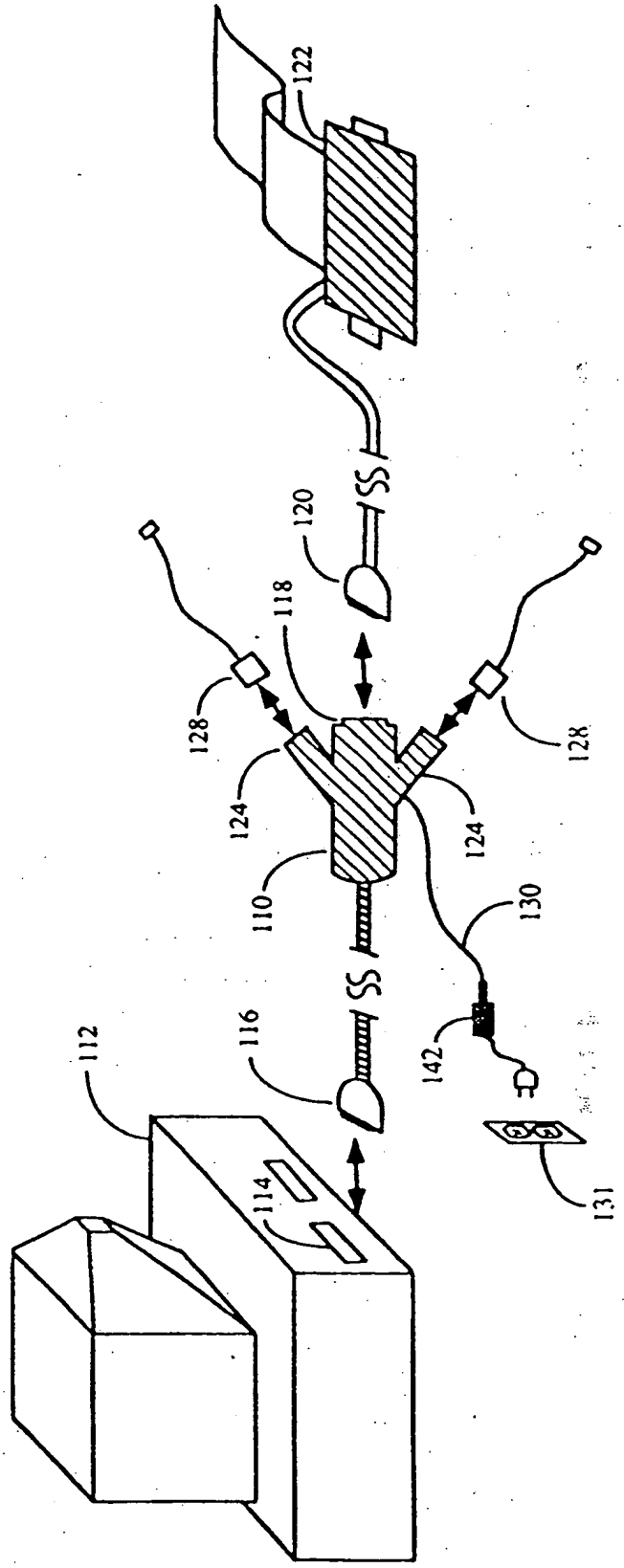
d.) a control unit coupled to said host interface, said peripheral interface, and said network interface, said control unit selectively coupling or uncoupling said host data lines and said host control lines with said peripheral data lines and said peripheral control lines, said control unit including a file manager coupled to said network interface and said host interface for controlling file transfers between said network and a host computer;

a first host computer coupled to said host interface;

a second host computer coupled to said network interface; and

a peripheral device coupled to said peripheral interface, said control unit further including control logic providing access of said peripheral device to said first host computer and said second host computer, said file manager providing control logic for transferring a file between said first host computer and said second host computer.

FIGURE 1



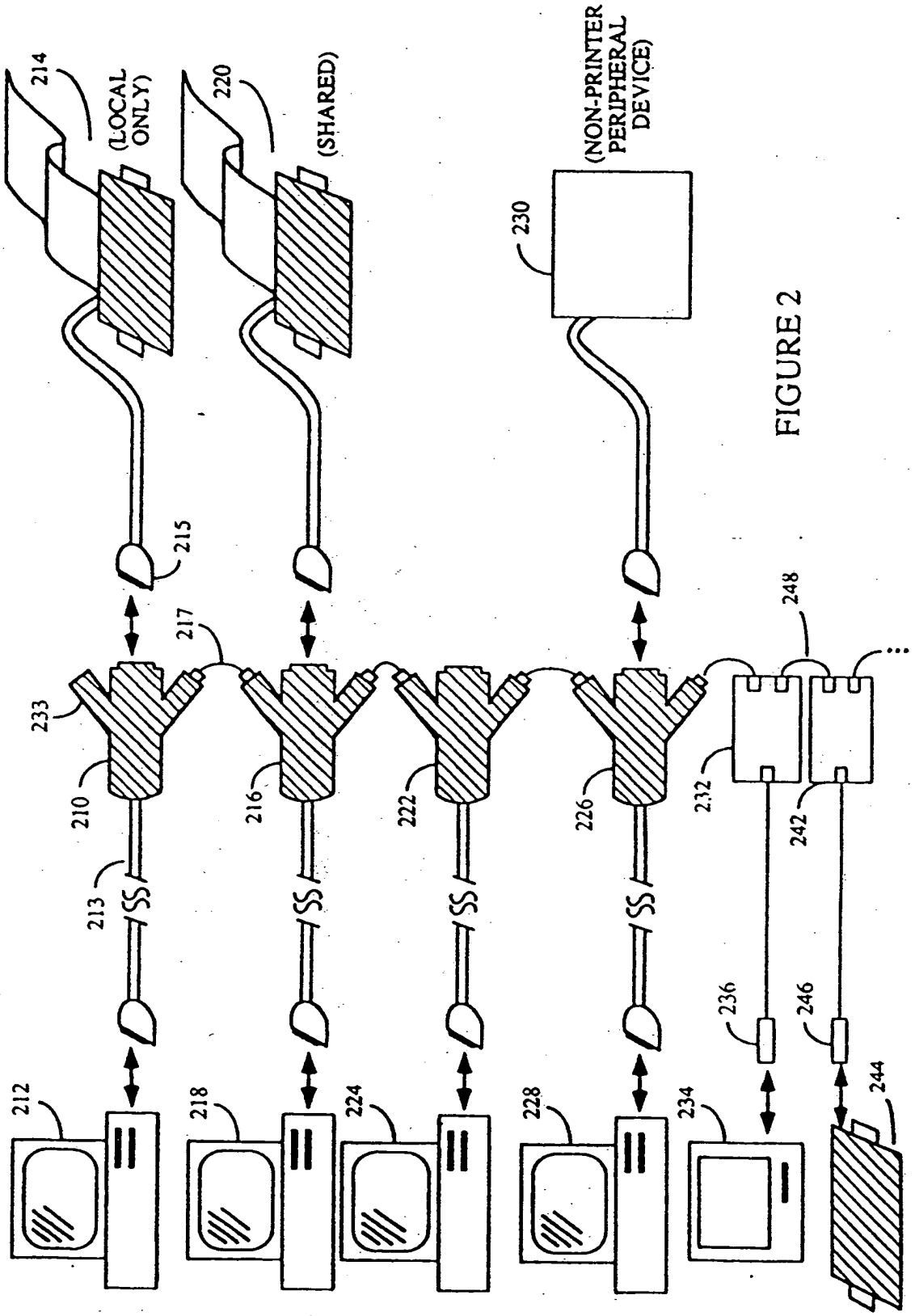
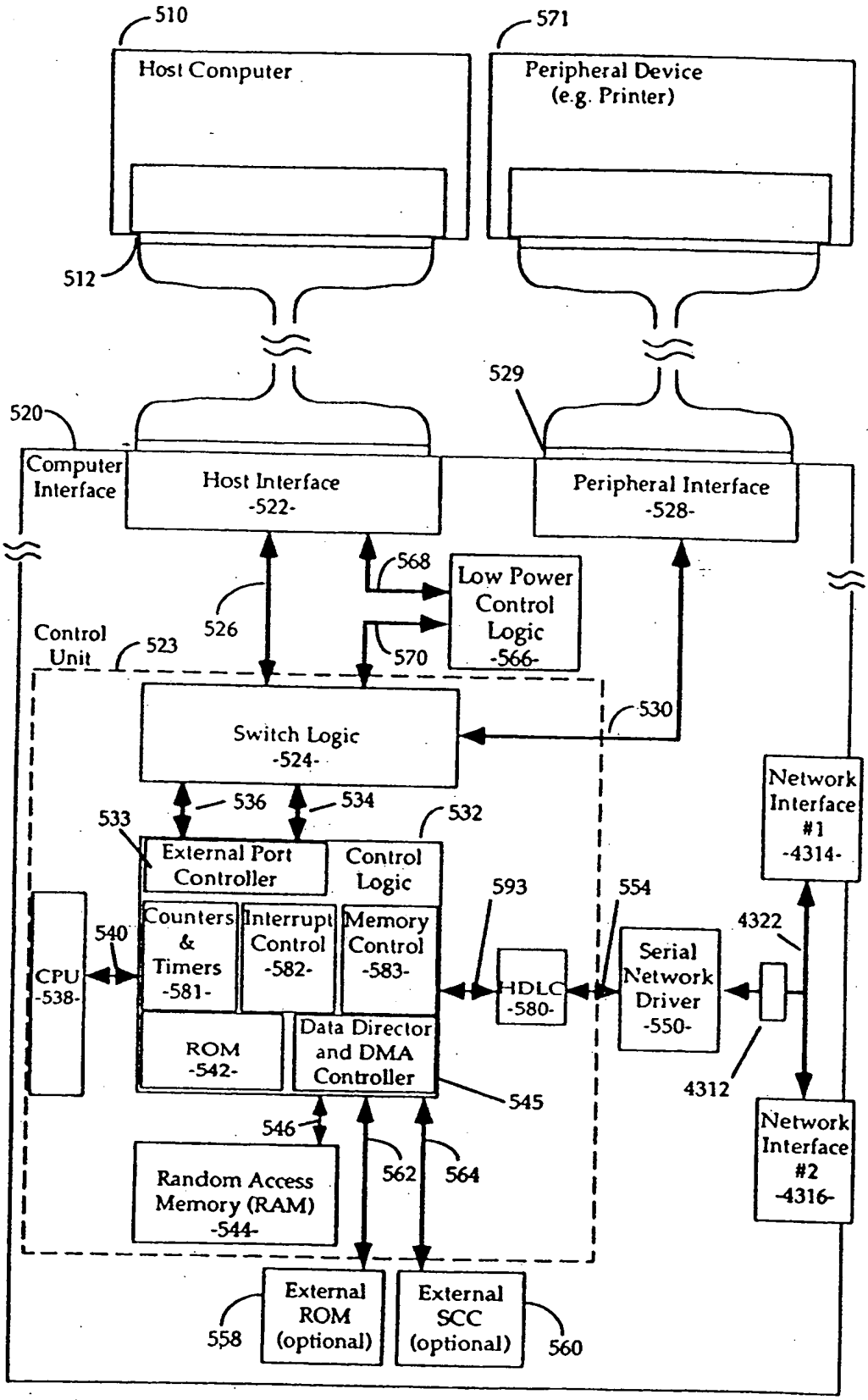

214
(LOCAL ONLY)
220
(SHARED)
(NON-PRINTER PERIPHERAL DEVICE)
230
215
217
248
233
210
216
222
226
232
242
213
236
246
212
218
224
228
234
244

FIGURE 2

FIGURE 3



SUBSTITUTE SHEET (RULE 26)

FIGURE 4A

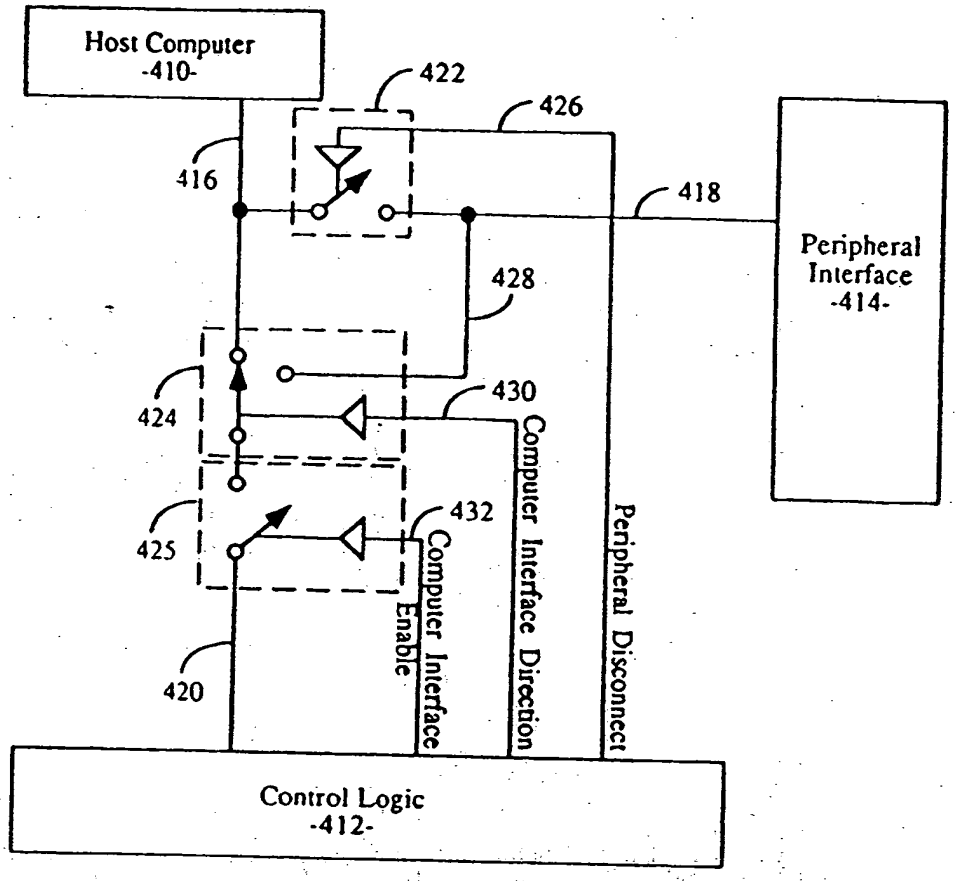

Host Computer
-410-
422
426
416
418
Peripheral
Interface
-414-
428
424
430
Computer Interface Direction
Peripheral Disconnect
425
432
Computer Interface
Enable
420
Control Logic
-412-

FIGURE 4B

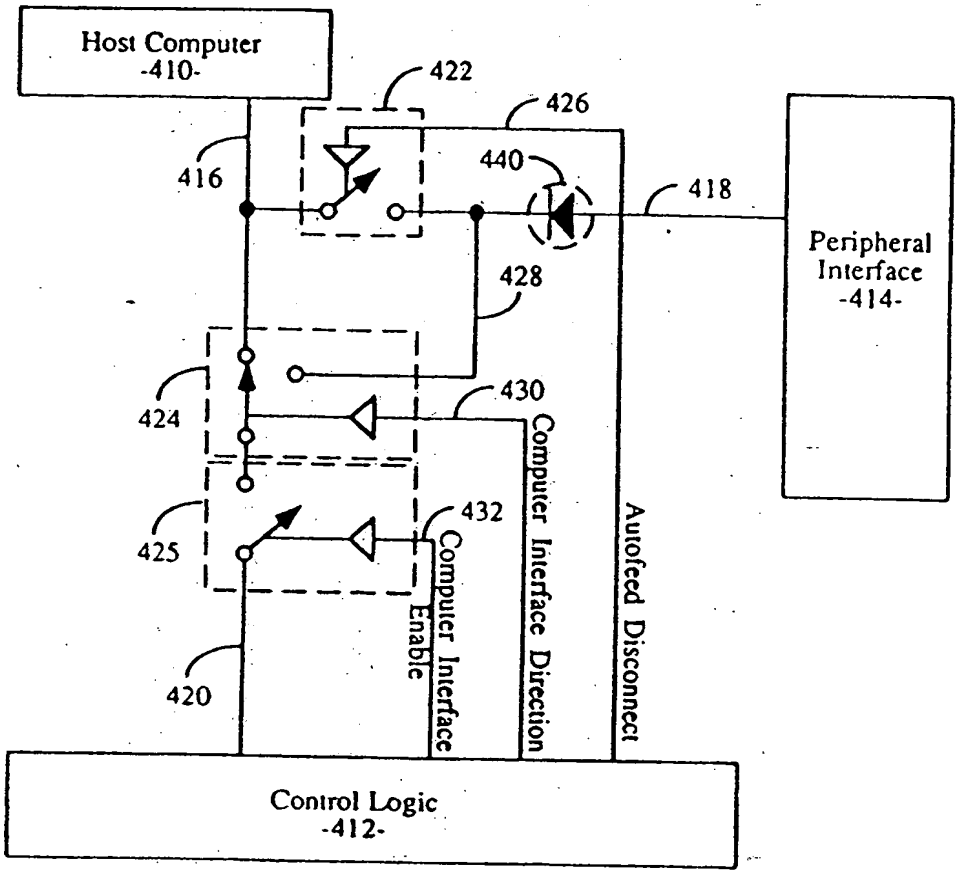
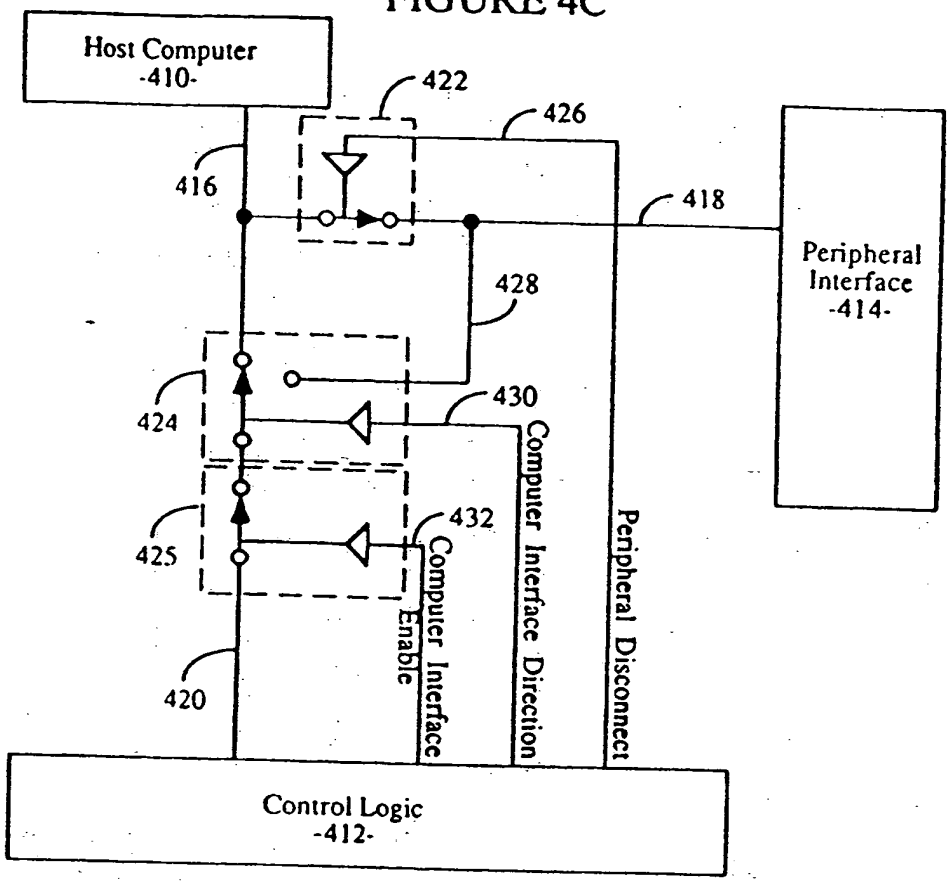


FIGURE 4C


Host Computer
-410-
422
426
416
418
Peripheral
Interface
-414-
428
424
430
Computer Interface Direction
425
432
Computer Interface
Enable
Peripheral Disconnect
420
Control Logic
-412-

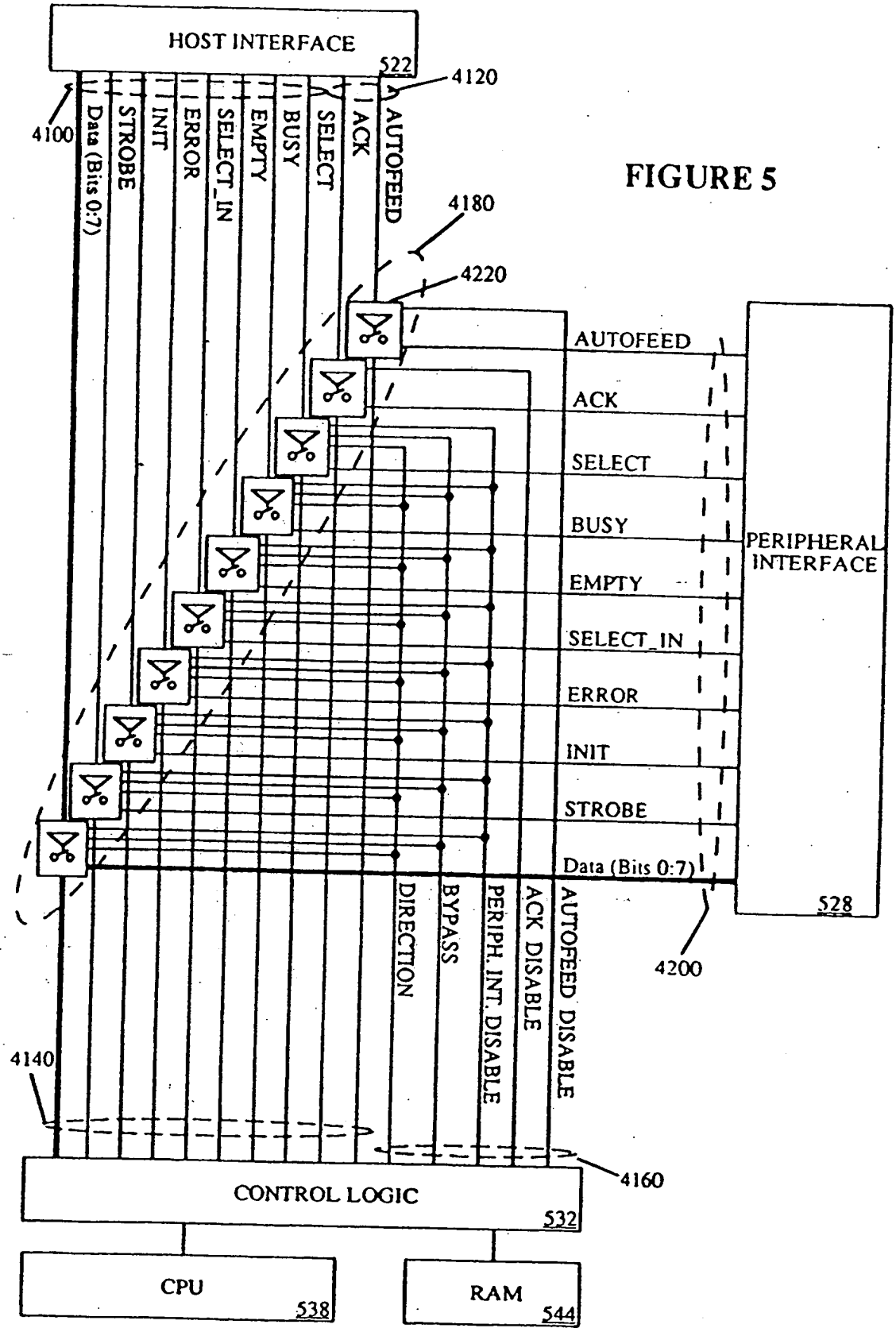


FIGURE 5

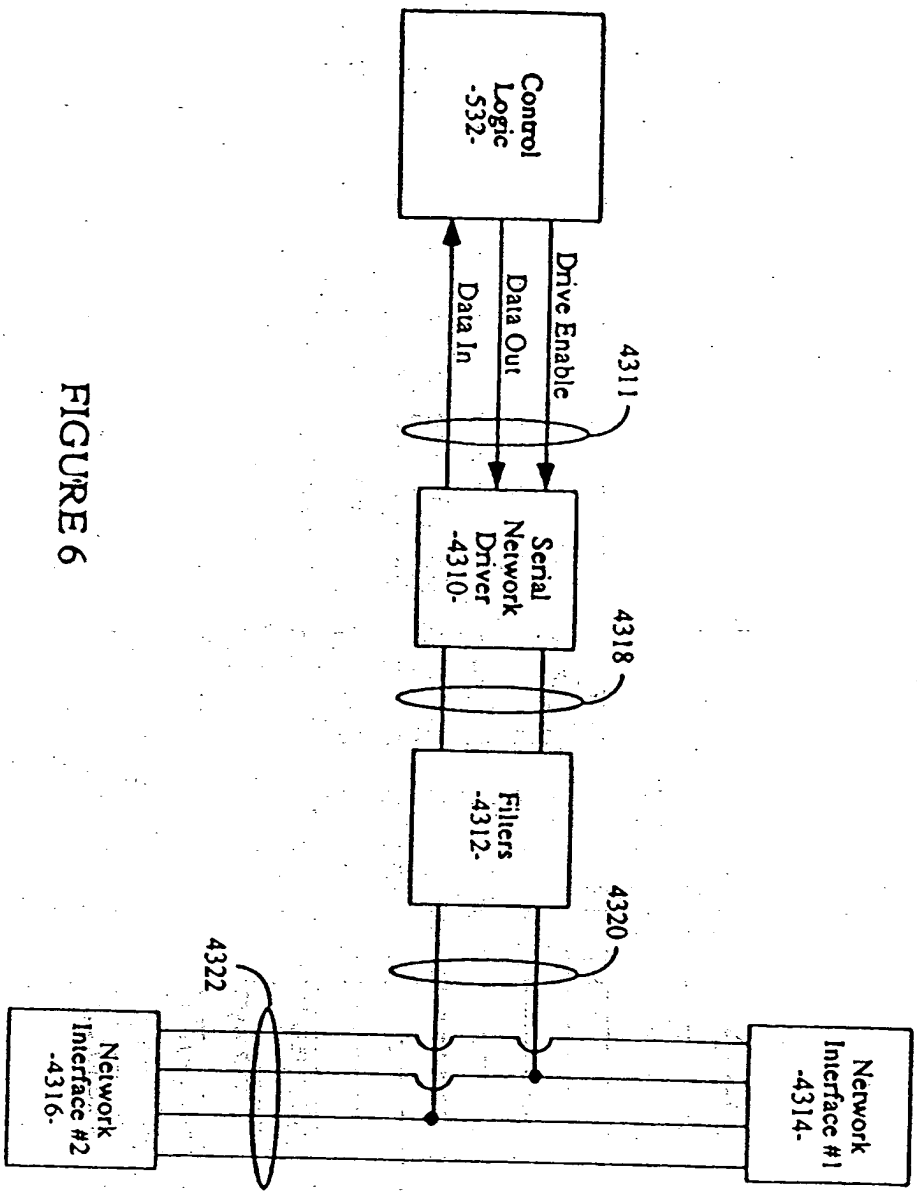

Control Logic -532-
Drive Enable
Data Out
Data In
4311
Serial Network Driver -4310-
4318
Filters -4312-
4320
4322
Network Interface #2 -4316-
Network Interface #1 -4314-

FIGURE 6

FIGURE 7

Acknowledge Control/Status Register [ACK]

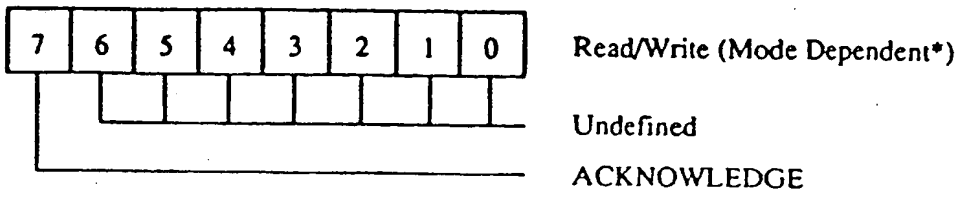


FIGURE 8

Autofeed Control/Status Register [AF]

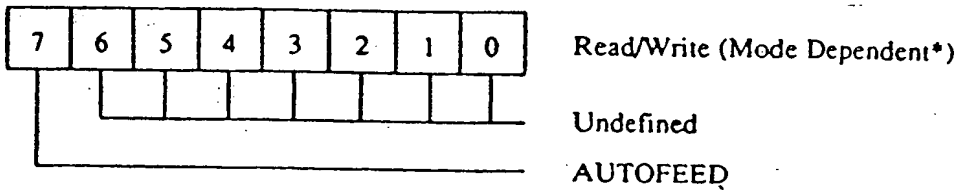


FIGURE 9

Switching Control/Status Register [SW_CTRL]

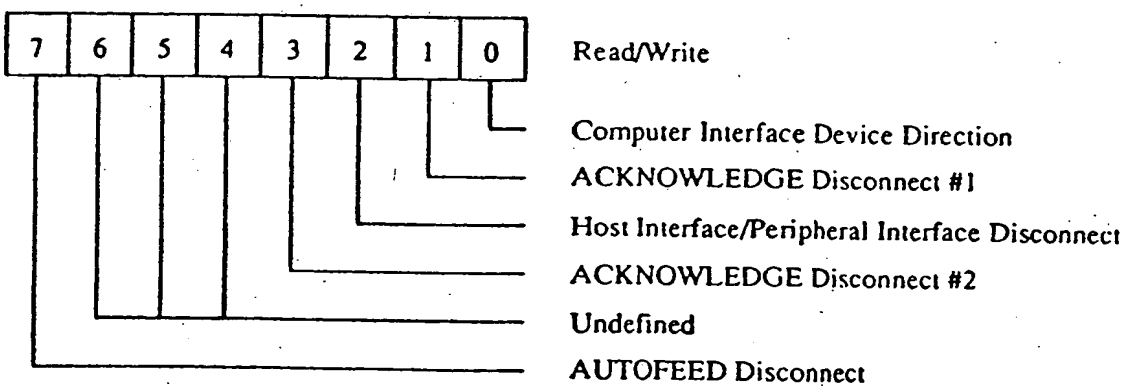


FIGURE 10

External Port Data Register [EPDATA]

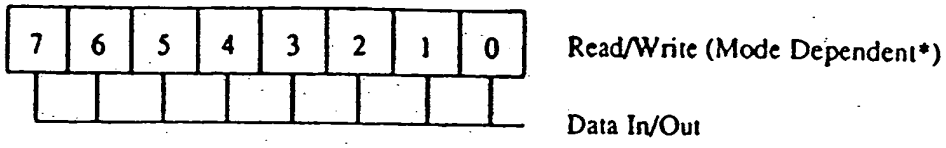


FIGURE 11

External Port Status Register [PRN_STAT:TXDATA]

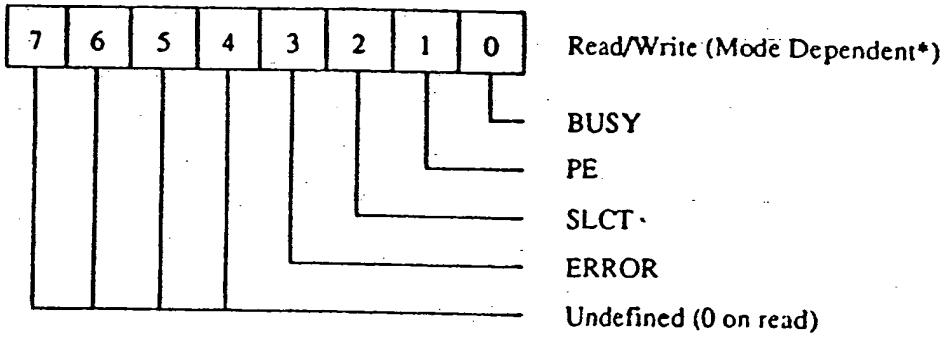


FIGURE 12

Printer Control (Strobes) Register [PRN_CTRL]

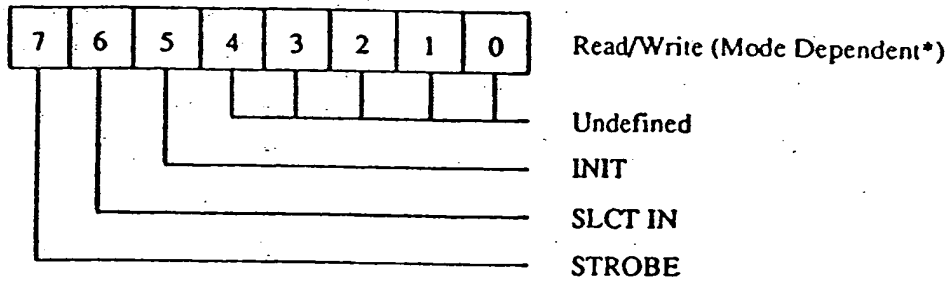


FIGURE 13

Switching Output Control Register [SWOE]

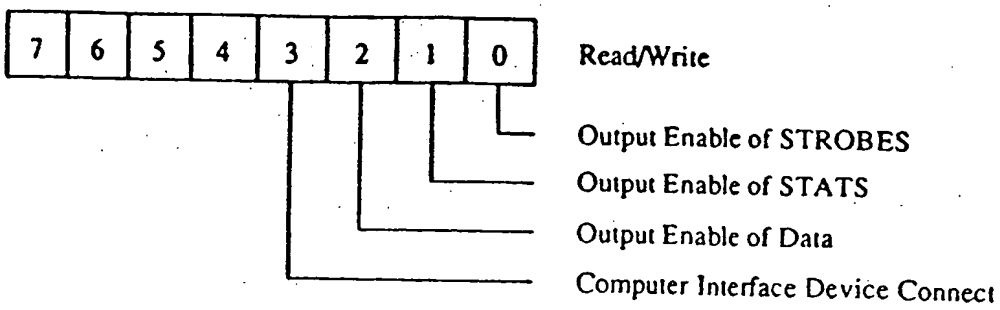


FIGURE 14

External Port Mode Register [EPMR]

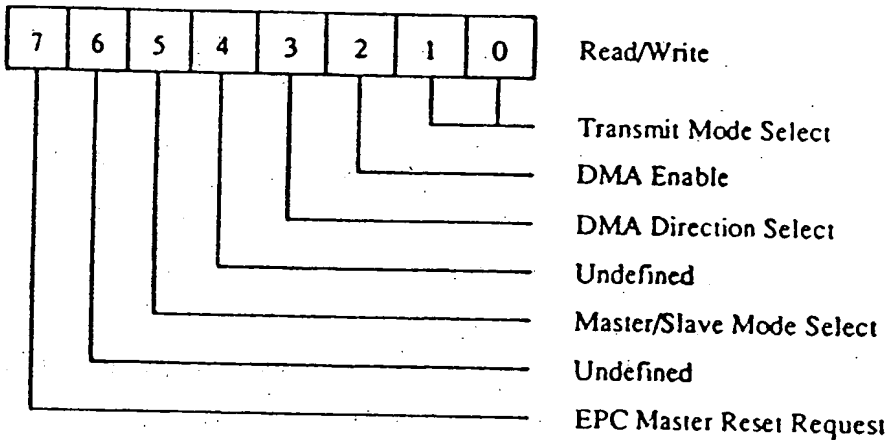


FIGURE 15

BIT 1	BIT 0	Mode Definition
0	0	2 Bit Mode (Tandy) (default)
0	1	4 Bit Mode (XT)
1	0	8 Bit Mode (PS/2)
1	1	Undefined

FIGURE 16

External Port Interrupt Enable Register [EPINTEN]

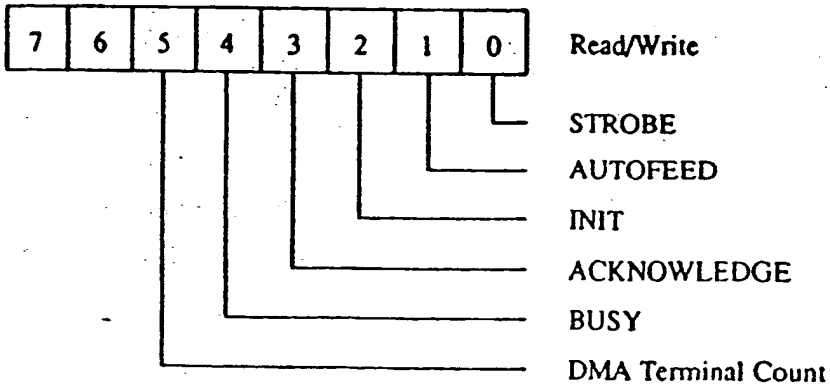


FIGURE 17

External Port Interrupt Status Register [EPINTST]

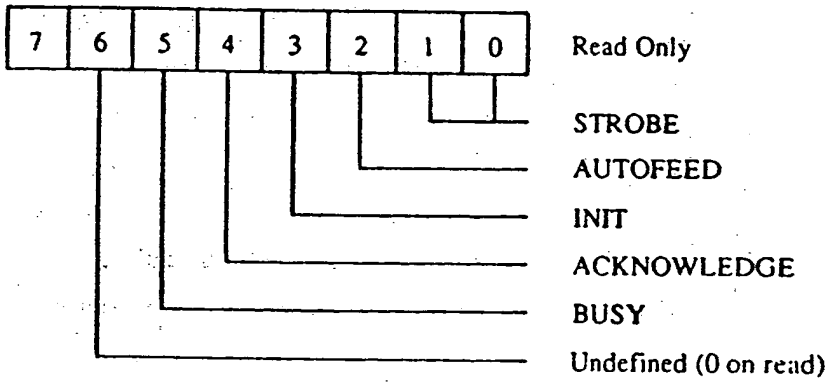


FIGURE 18

External Port Checksum Register [CHKSUM]

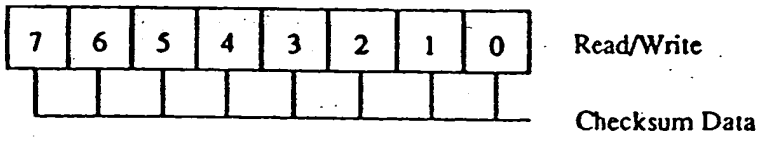


FIGURE 19

Mode	Xfer #	Content	Mux 1	Mux 2	Mux 3	Mux 4
Slave Modes						
2 Bit	1	7:6	1	1	0	X
	2	5:4	1	0	0	X
	3	3:2	0	1	0	X
	4	1:0	0	0	0	X
4 Bit	1	7:4	1	0	0	X
	2	3:0	0	0	0	X
8 Bit	1	7:0	X	X	X	X
POLLED	X	Registered	X	X	X	X
Master Modes						
8 Bit (Polled)	1	7:0	X	X	X	1
8 Bit (Computer Interface Device to Computer Interface Device)	1	7:0	X	X	X	0

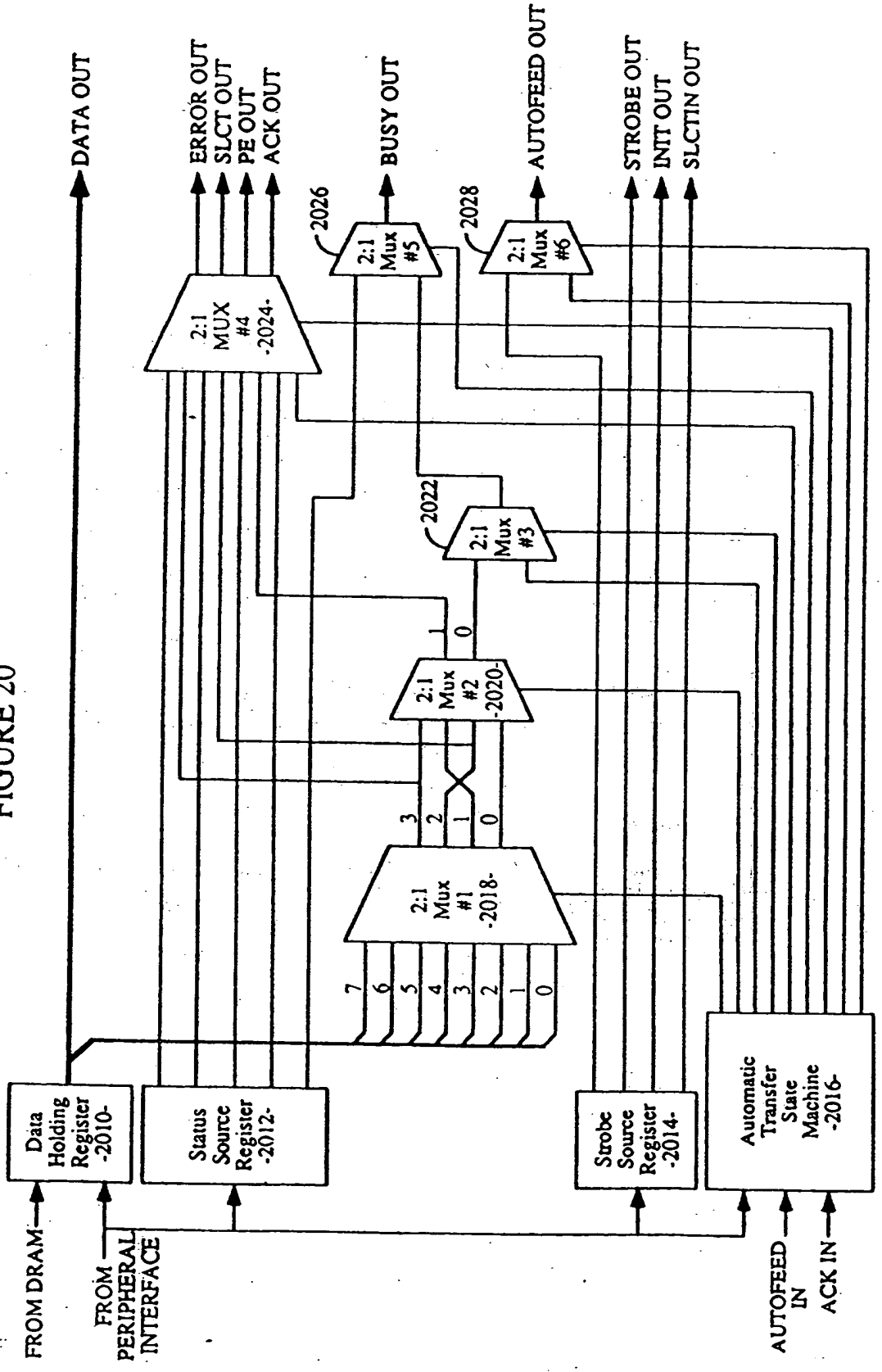

DATA OUT
ERROR OUT
SLCT OUT
PE OUT
ACK OUT
BUSY OUT
AUTOFEED OUT
STROBE OUT
INIT OUT
SLCTIN OUT
2026
2028
2022
2:1 Mux #5
2:1 Mux #6
2:1 MUX #4 -2024-
2:1 Mux #3
2:1 Mux #2 -2020-
2:1 Mux #1 -2018-
Data Holding Register -2010-
Status Source Register -2012-
Strobe Source Register -2014-
Automatic Transfer State Machine -2016-
FROM DRAM
FROM PERIPHERAL INTERFACE
AUTOFEED IN
ACK IN

FIGURE 20

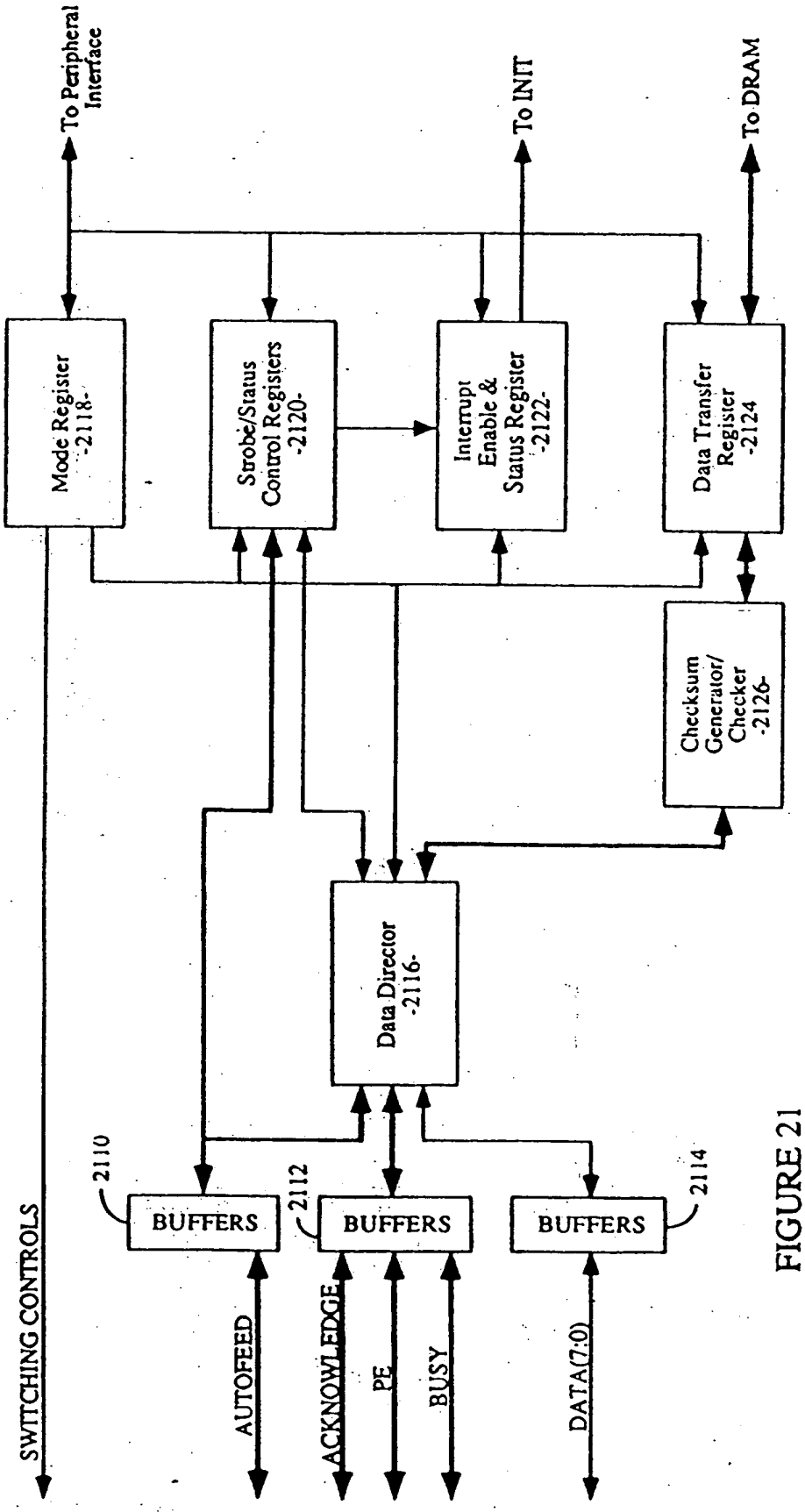


FIGURE 21

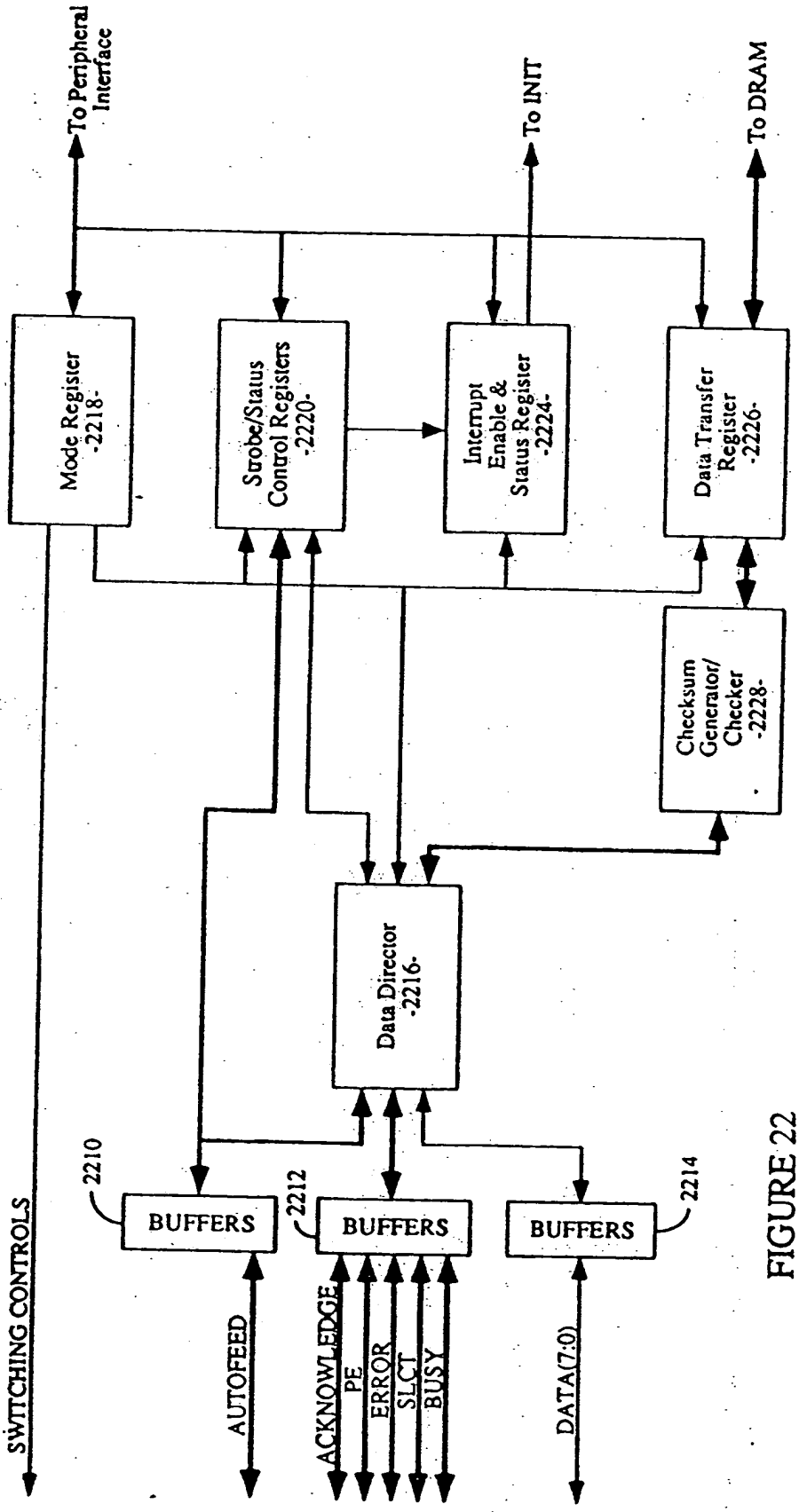

To Peripheral Interface
To INIT
To DRAM
Mode Register -2218-
Strobe/Staus Control Registers -2220-
Interrupt Enable & Status Register -2224-
Data Transfer Register -2226-
Checksum Generator/ Checker -2228-
Data Director -2216-
2210
2212
2214
BUFFERS
BUFFERS
BUFFERS
SWITCHING CONTROLS
AUTOFEED
ACKNOWLEDGE
PE
ERROR
SLCT
BUSY
DATA(7:0)

FIGURE 22

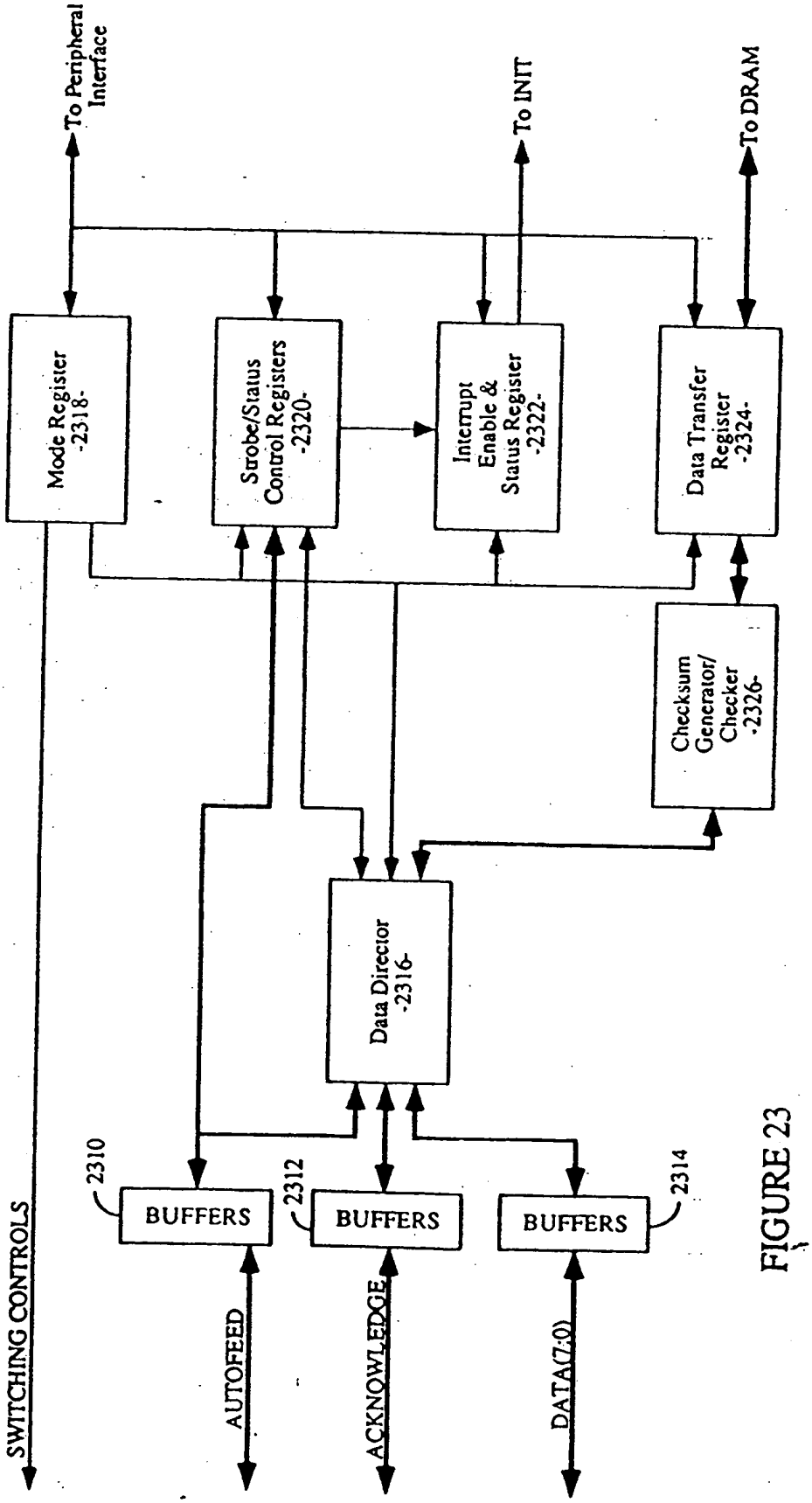
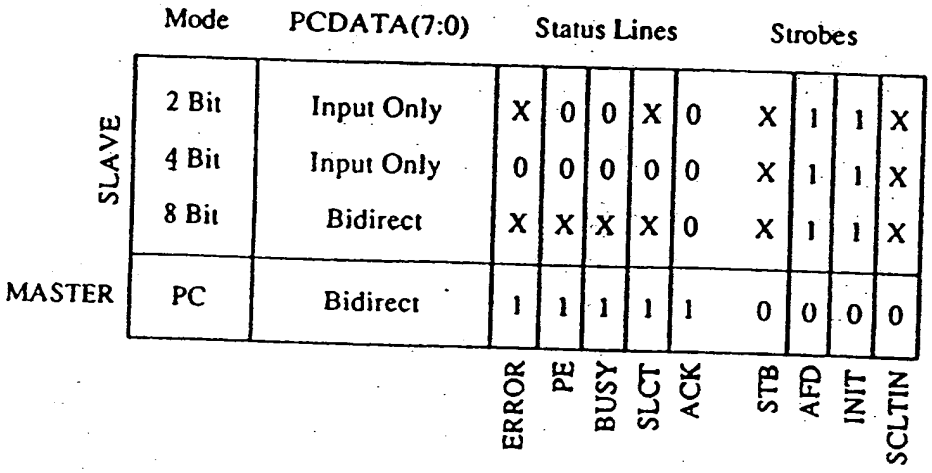


FIGURE 23

FIGURE 24



	Mode	PCDATA(7:0)	ERROR	PE	BUSY	SLCT	ACK	STB	AFD	INIT	SCLTIN
SLAVE	2 Bit	Input Only	X	0	0	X	0	X	1	1	X
SLAVE	4 Bit	Input Only	0	0	0	0	0	X	1	1	X
SLAVE	8 Bit	Bidirect	X	X	X	X	0	X	1	1	X
MASTER	PC	Bidirect	1	1	1	1	1	0	0	0	0

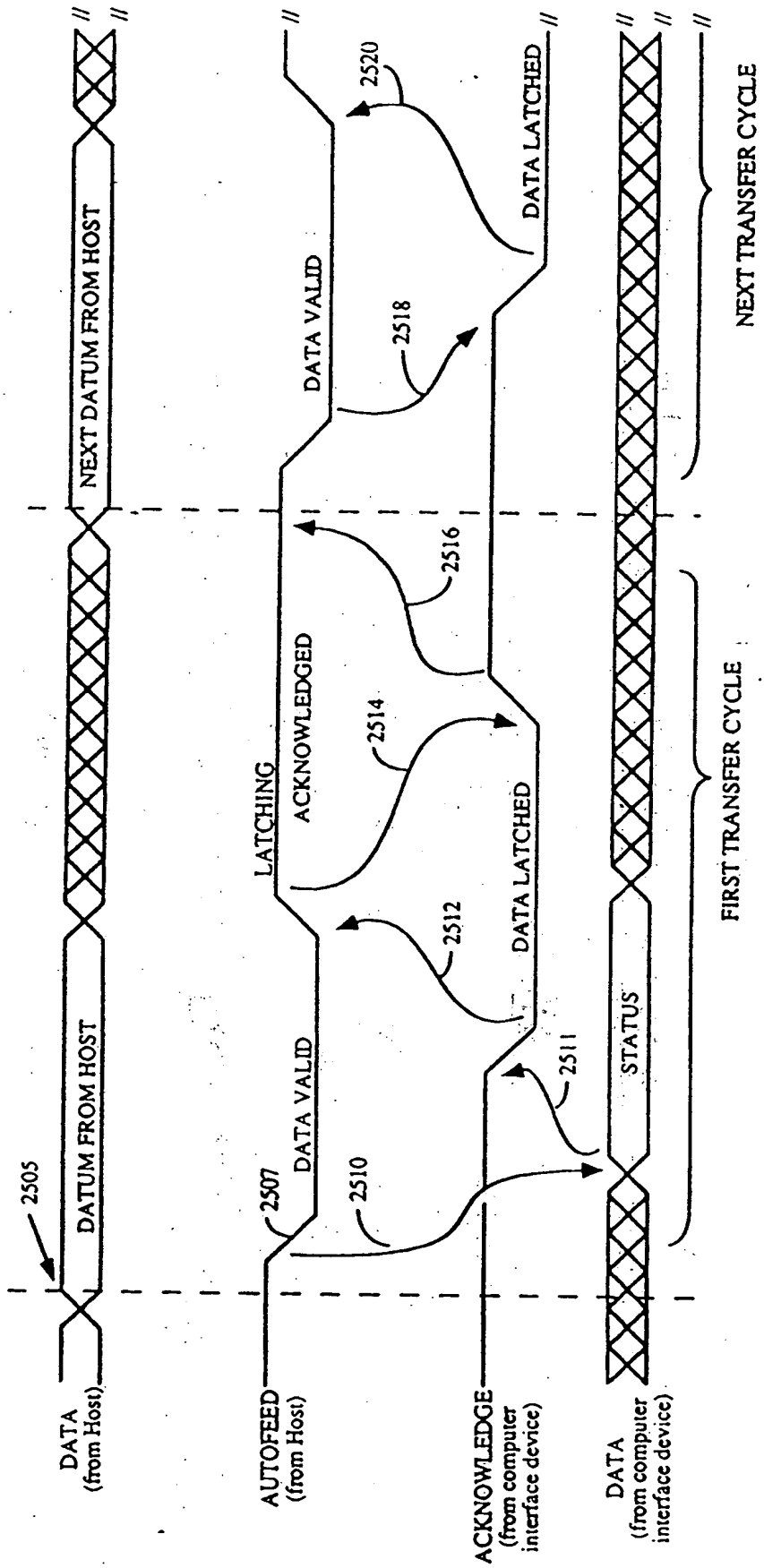


FIGURE 25

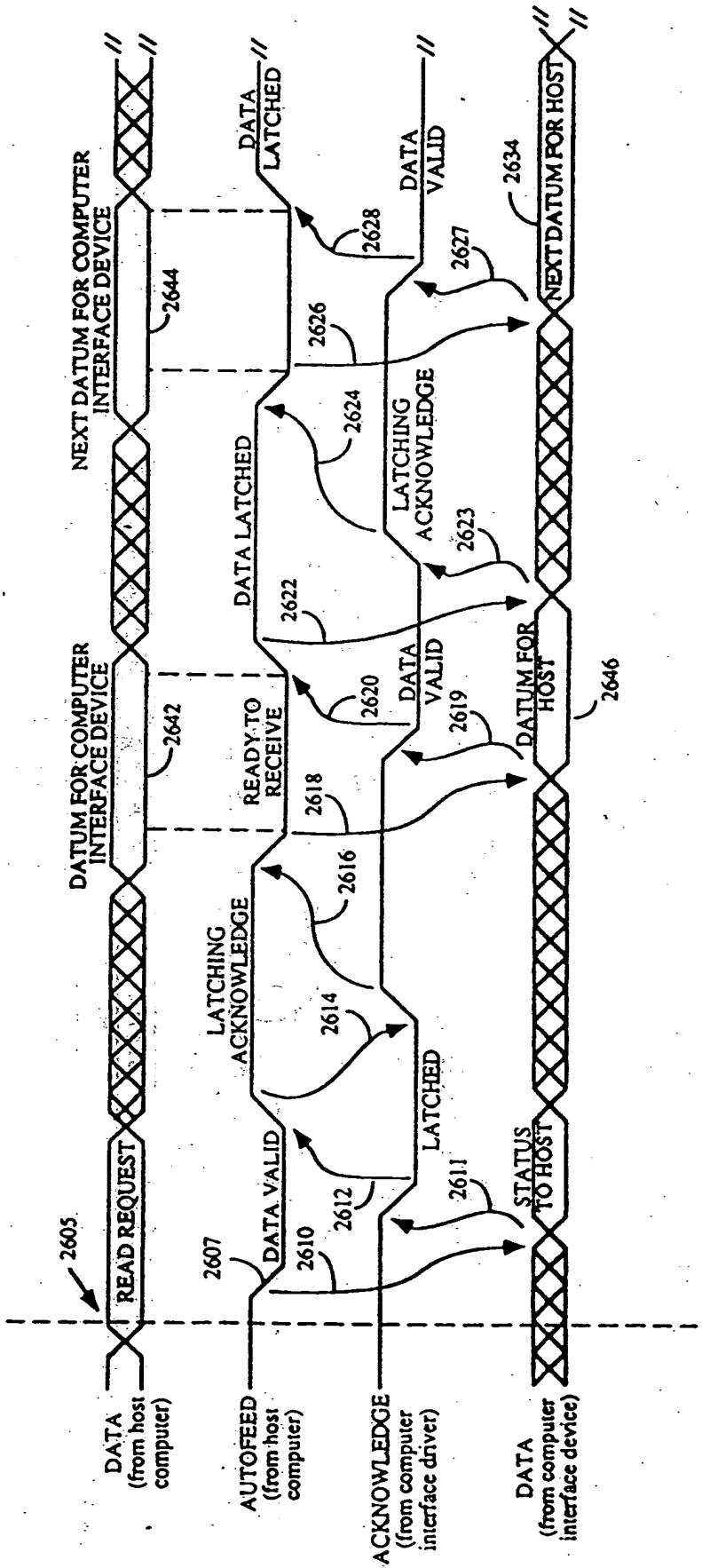

DATA (from host computer)
AUTOFEED (from host computer)
ACKNOWLEDGE (from computer interface driver)
DATA (from computer interface device)
2605
READ REQUEST
2607
DATA VALID
2610
2612
LATCHED
2611
STATUS TO HOST
LATCHING ACKNOWLEDGE
2614
2616
READY TO RECEIVE
2618
DATUM FOR COMPUTER INTERFACE DEVICE
2642
DATA VALID
2620
2619
DATUM FOR HOST
2646
DATA LATCHED
2622
LATCHING ACKNOWLEDGE
2624
2623
DATA VALID
2626
2628
2627
DATA LATCHED
NEXT DATUM FOR COMPUTER INTERFACE DEVICE
2644
NEXT DATUM FOR HOST
2634

FIGURE 26

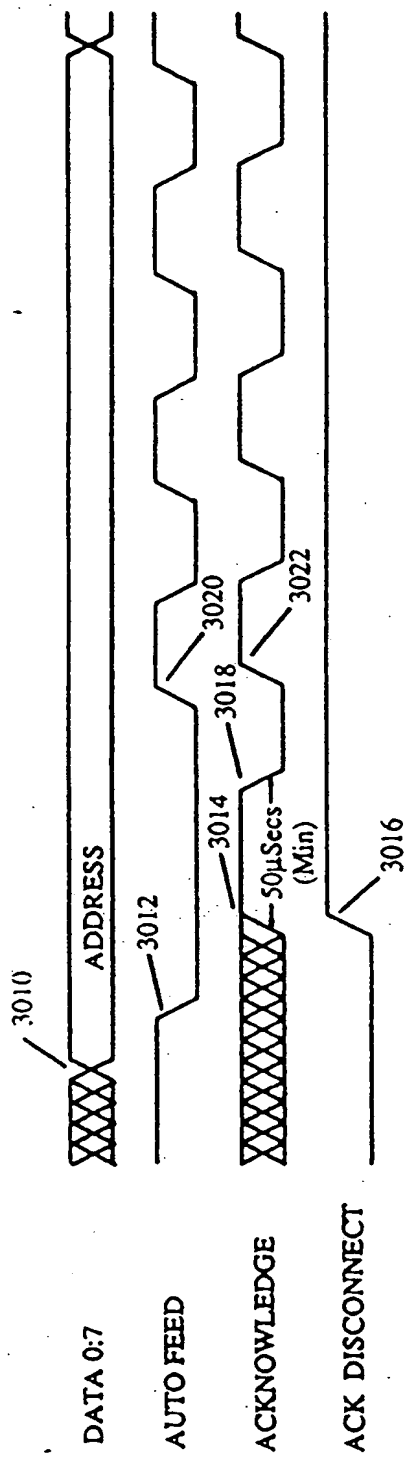


Figure 27

FIGURE 28

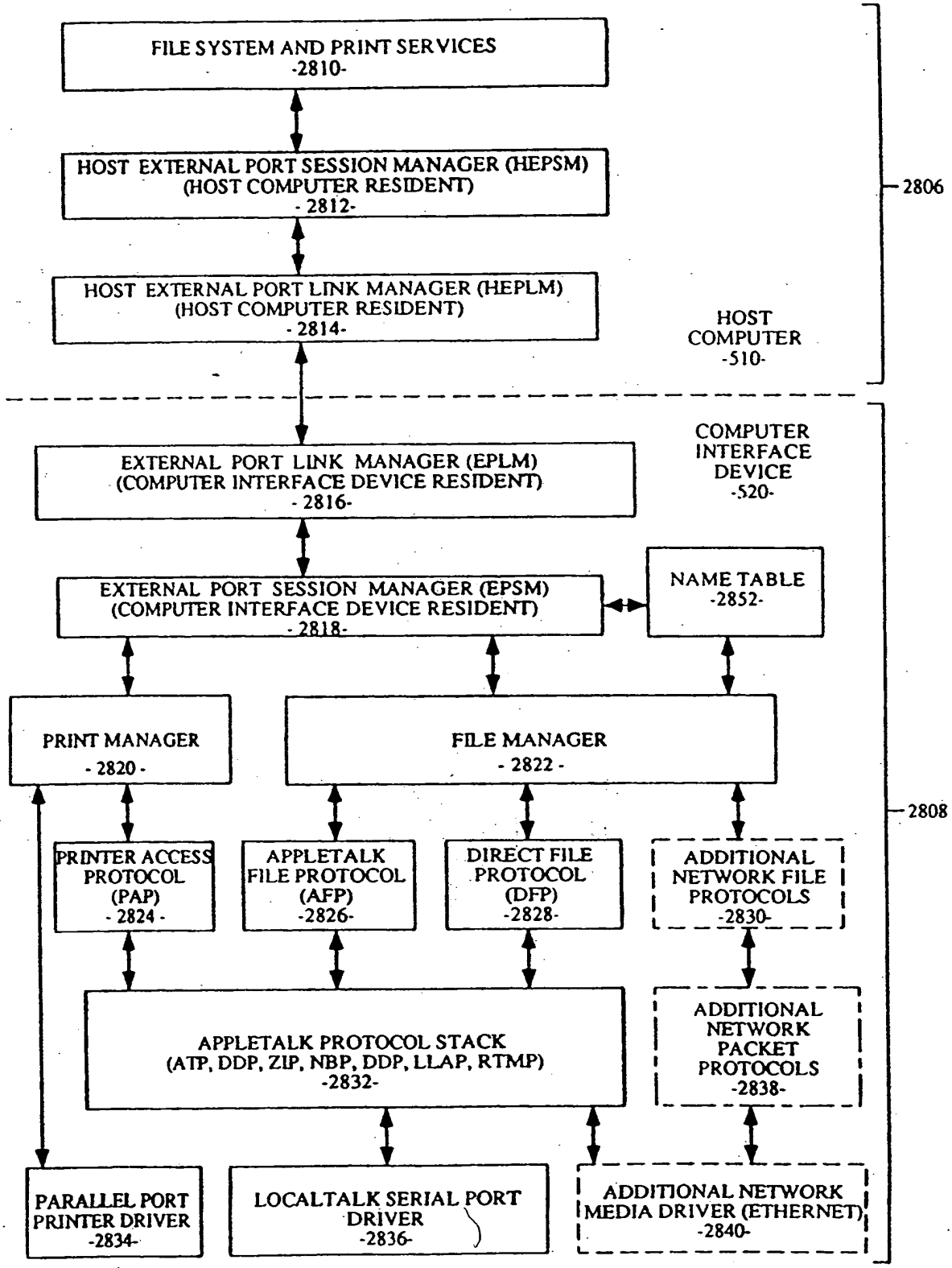

FILE SYSTEM AND PRINT SERVICES
-2810-
HOST EXTERNAL PORT SESSION MANAGER (HEPSM)
(HOST COMPUTER RESIDENT)
- 2812-
HOST EXTERNAL PORT LINK MANAGER (HEPLM)
(HOST COMPUTER RESIDENT)
- 2814-
2806
HOST
COMPUTER
-510-
EXTERNAL PORT LINK MANAGER (EPLM)
(COMPUTER INTERFACE DEVICE RESIDENT)
- 2816-
COMPUTER
INTERFACE
DEVICE
-520-
EXTERNAL PORT SESSION MANAGER (EPSM)
(COMPUTER INTERFACE DEVICE RESIDENT)
- 2818-
NAME TABLE
-2852-
PRINT MANAGER
- 2820 -
FILE MANAGER
- 2822 -
2808
PRINTER ACCESS
PROTOCOL
(PAP)
- 2824 -
APPLETALK
FILE PROTOCOL
(AFP)
-2826-
DIRECT FILE
PROTOCOL
(DFP)
-2828-
ADDITIONAL
NETWORK FILE
PROTOCOLS
-2830-
APPLETALK PROTOCOL STACK
(ATP, DDP, ZIP, NBP, DDP, LLAP, RTMP)
-2832-
ADDITIONAL
NETWORK
PACKET
PROTOCOLS
-2838-
PARALLEL PORT
PRINTER DRIVER
-2834-
LOCALTALK SERIAL PORT
DRIVER
-2836-
ADDITIONAL NETWORK
MEDIA DRIVER (ETHERNET)
-2840-

FIGURE 29

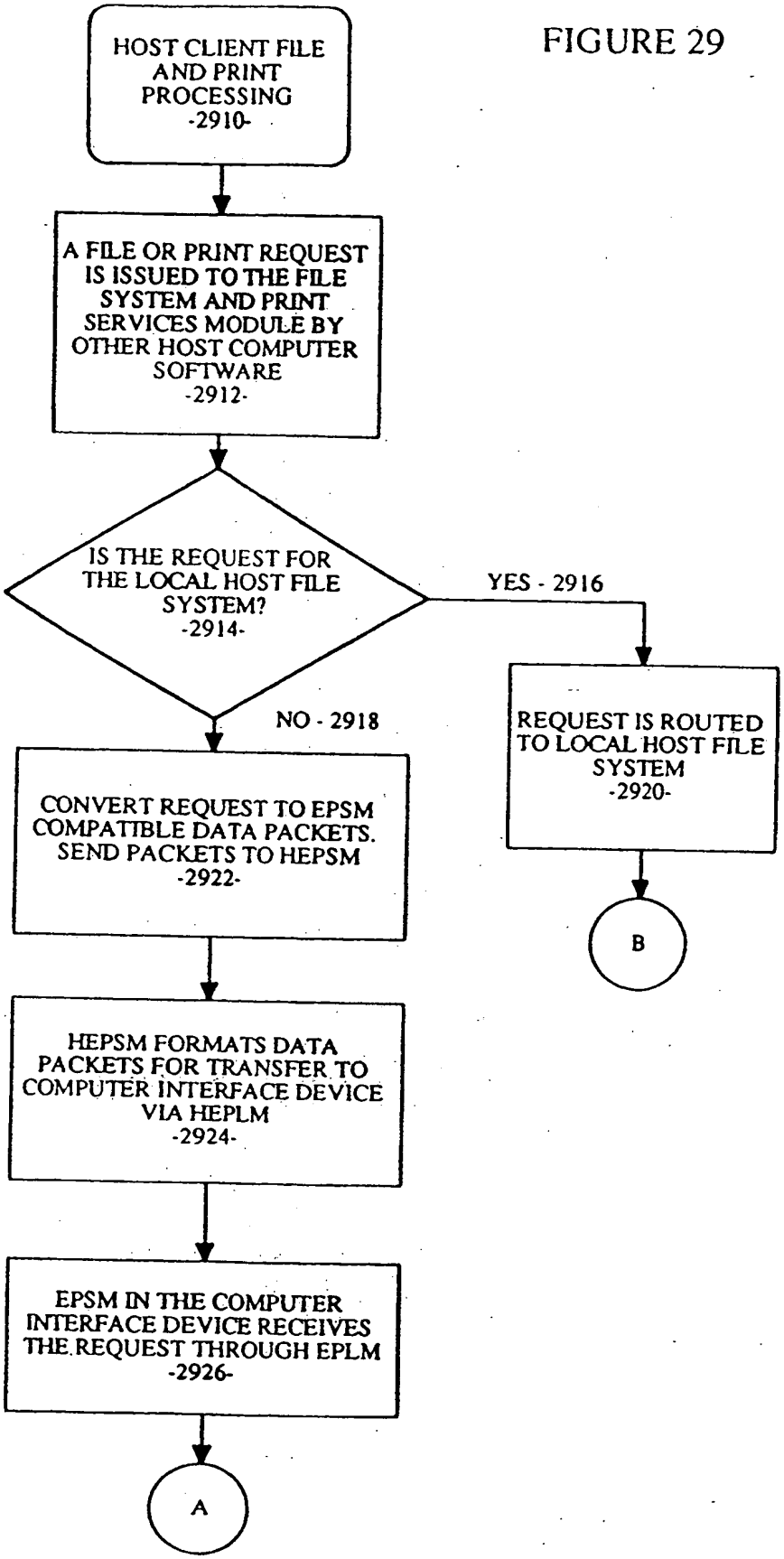


FIGURE 30

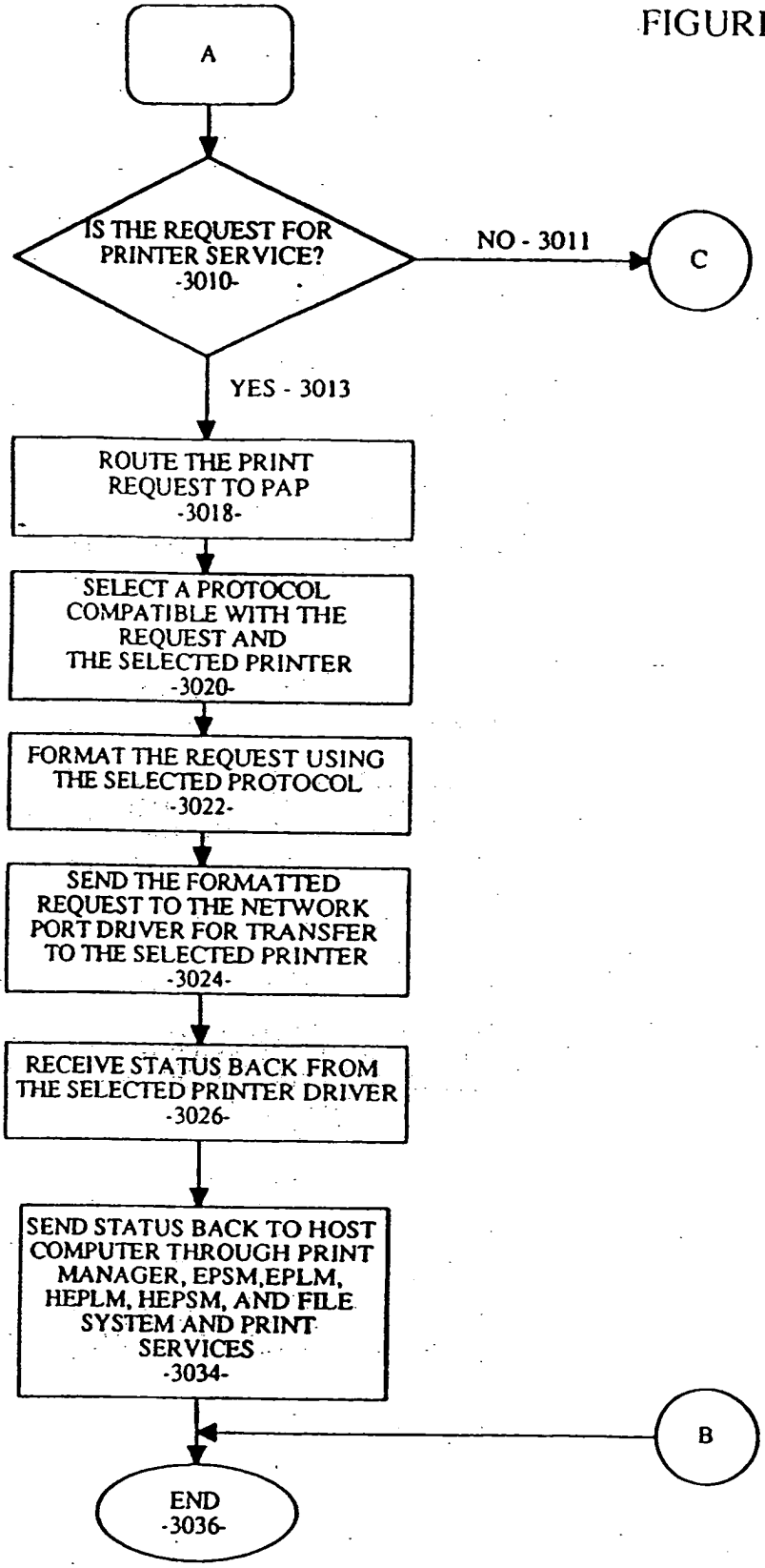

A
IS THE REQUEST FOR PRINTER SERVICE? -3010-
NO - 3011
C
YES - 3013
ROUTE THE PRINT REQUEST TO PAP -3018-
SELECT A PROTOCOL COMPATIBLE WITH THE REQUEST AND THE SELECTED PRINTER -3020-
FORMAT THE REQUEST USING THE SELECTED PROTOCOL -3022-
SEND THE FORMATTED REQUEST TO THE NETWORK PORT DRIVER FOR TRANSFER TO THE SELECTED PRINTER -3024-
RECEIVE STATUS BACK FROM THE SELECTED PRINTER DRIVER -3026-
SEND STATUS BACK TO HOST COMPUTER THROUGH PRINT MANAGER, EPSM,EPLM, HEPLM, HEPSM, AND FILE SYSTEM AND PRINT SERVICES -3034-
B
END -3036-

FIGURE 31

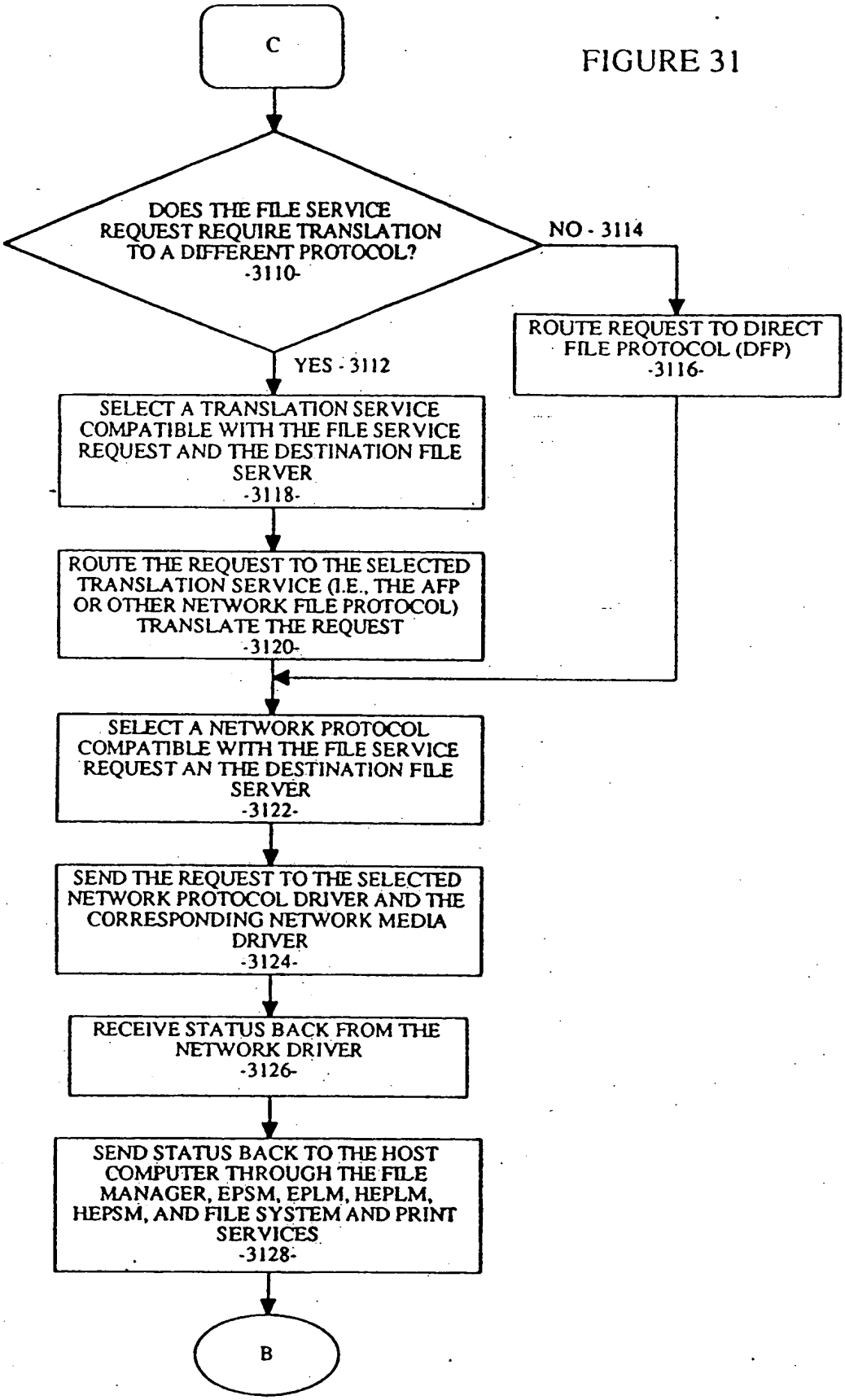

C
DOES THE FILE SERVICE REQUEST REQUIRE TRANSLATION TO A DIFFERENT PROTOCOL? -3110-
NO - 3114
ROUTE REQUEST TO DIRECT FILE PROTOCOL (DFP) -3116-
YES - 3112
SELECT A TRANSLATION SERVICE COMPATIBLE WITH THE FILE SERVICE REQUEST AND THE DESTINATION FILE SERVER -3118-
ROUTE THE REQUEST TO THE SELECTED TRANSLATION SERVICE (I.E., THE AFP OR OTHER NETWORK FILE PROTOCOL) TRANSLATE THE REQUEST -3120-
SELECT A NETWORK PROTOCOL COMPATIBLE WITH THE FILE SERVICE REQUEST AN THE DESTINATION FILE SERVER -3122-
SEND THE REQUEST TO THE SELECTED NETWORK PROTOCOL DRIVER AND THE CORRESPONDING NETWORK MEDIA DRIVER -3124-
RECEIVE STATUS BACK FROM THE NETWORK DRIVER -3126-
SEND STATUS BACK TO THE HOST COMPUTER THROUGH THE FILE MANAGER, EPSM, EPLM, HEPLM, HEPSM, AND FILE SYSTEM AND PRINT SERVICES -3128-
B

FIGURE 32

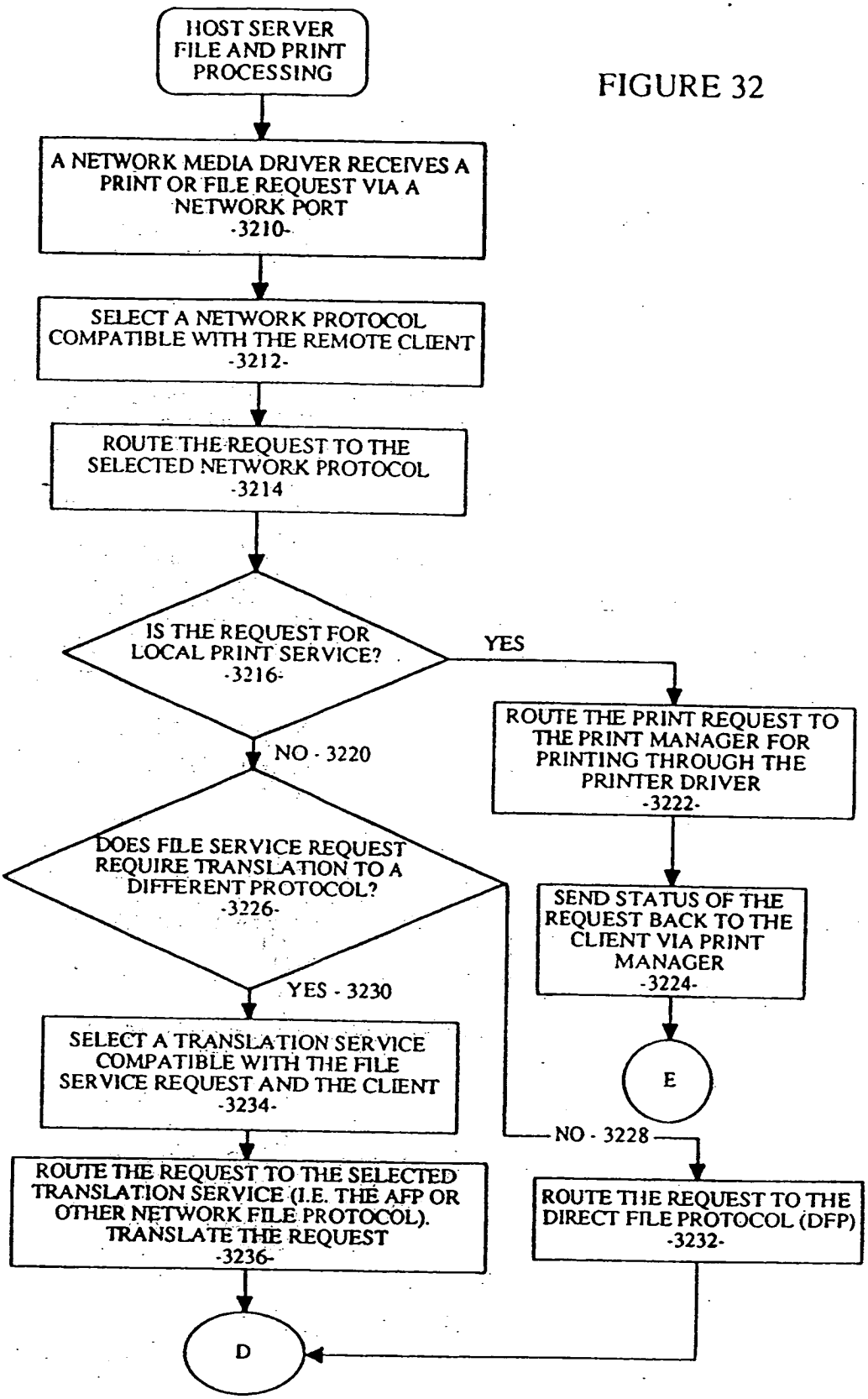
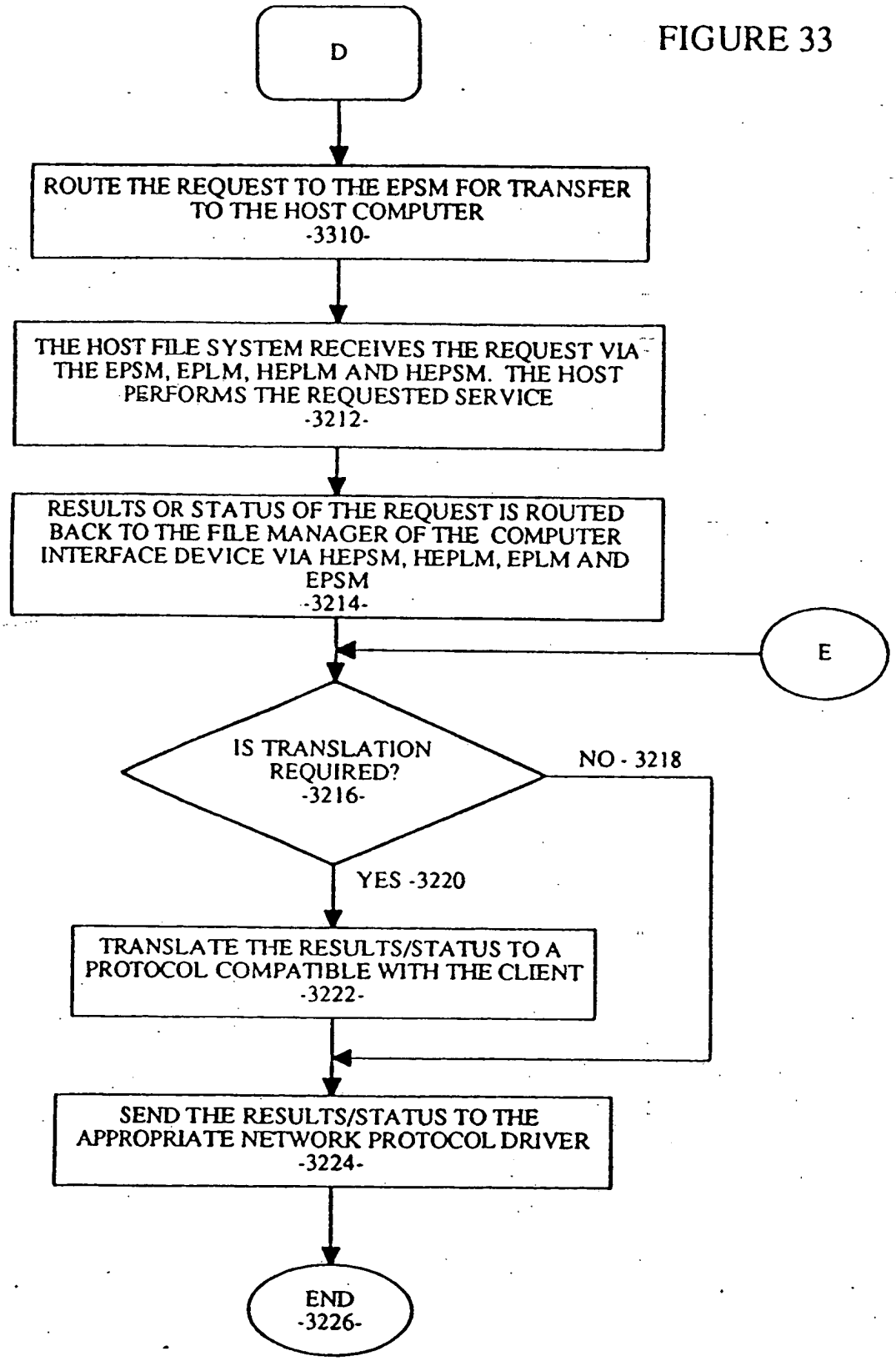

HOST SERVER FILE AND PRINT PROCESSING
A NETWORK MEDIA DRIVER RECEIVES A PRINT OR FILE REQUEST VIA A NETWORK PORT -3210-
SELECT A NETWORK PROTOCOL COMPATIBLE WITH THE REMOTE CLIENT -3212-
ROUTE THE REQUEST TO THE SELECTED NETWORK PROTOCOL -3214
IS THE REQUEST FOR LOCAL PRINT SERVICE? -3216-
YES
ROUTE THE PRINT REQUEST TO THE PRINT MANAGER FOR PRINTING THROUGH THE PRINTER DRIVER -3222-
SEND STATUS OF THE REQUEST BACK TO THE CLIENT VIA PRINT MANAGER -3224-
E
NO - 3220
DOES FILE SERVICE REQUEST REQUIRE TRANSLATION TO A DIFFERENT PROTOCOL? -3226-
YES - 3230
SELECT A TRANSLATION SERVICE COMPATIBLE WITH THE FILE SERVICE REQUEST AND THE CLIENT -3234-
NO - 3228
ROUTE THE REQUEST TO THE SELECTED TRANSLATION SERVICE (I.E. THE AFP OR OTHER NETWORK FILE PROTOCOL). TRANSLATE THE REQUEST -3236-
ROUTE THE REQUEST TO THE DIRECT FILE PROTOCOL (DFP) -3232-
D

FIGURE 33



INTERNATIONAL SEARCH REPORT

Int tional application No.
PCT/US94/00031

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :Please See Extra Sheet.
US CL :395/275

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/275, 200

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
search terms: interface, peripheral, control lines, computer

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 4,106,092 (Millers, II.) 08 Aug. 1978, col. 5-8, 17-18, 23-26, 44-45, 67-68, 84-88.	1-8, 13-24

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be part of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

09 March 1994

Date of mailing of the international search report

APR 20 1994

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. NOT APPLICABLE

Authorized officer

DALE SHAW

B. Harold for

Telephone No. (703) 305-9717

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

Int ational application No.
PCT/US94/00031

A. CLASSIFICATION OF SUBJECT MATTER:
IPC (5):

G06F 13/00
G06F 13/00